TC2100 PK2

Organization_____ Bldg./Room_____

U. S. DEPARTMENT OF COMMERCE
COMMISSIONER FOR PATENTS
P.O. BOX 1450
ALEXANDRIA, VA 22313-1450
IF UNDELIVERABLE RETURN IN TEN DAYS

OFFICIAL BUSINESS

AN EQUAL OPPORTU

WASHINGTON JUL 22 04 DC

U.S. OFFICIAL MAIL
PENALTY FOR PRIVATE USE $300
PB METER 7251870
=3.95=
U.S. POSTAGE

12

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/777,505	02/05/2001	Yevgeniy Shteyn	US018008	8371

7590 07/22/2004

Corporate Patent Counsel
Philips Electronics North America Corporation
580 White Plains Road
Tarrytown, NY 10591

EXAMINER	
HOSSAIN, TANIM M	

ART UNIT	PAPER NUMBER
2141	7

DATE MAILED: 07/22/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

USPTO TO PROVIDE ELECTRONIC ACCESS TO CITED U.S. PATENT REFERENCES WITH OFFICE ACTIONS AND CEASE SUPPLYING PAPER COPIES

In support of its 21st Century Strategic Plan goal of increased patent e-Government, beginning in June 2004, the United States Patent and Trademark Office (Office or USPTO) will begin the phase-in of its E-Patent Reference program and hence will: (1) **provide downloading capability of the U.S. patents and U.S. patent application publications cited in Office actions** via the E-Patent Reference feature of the Office's Patent Application Information Retrieval (PAIR) system; and (2) **cease mailing paper copies of U.S. patents and U.S. patent application publications with Office actions** (in applications and during reexamination proceedings) except for citations made during the international stage of an international application under the Patent Cooperation Treaty (PCT). In order to use the new E-Patent Reference feature applicants must: (1) obtain a digital certificate and software from the Office; (2) obtain a customer number from the Office; and (3) properly associate patent applications with the customer number. Alternatively, copies of all U.S. patents and patent application publications can be accessed without a digital certificate from the USPTO web site, from the USPTO Office of Public Records, and from commercial sources. The Office will continue the practice of supplying paper copies of foreign patent documents and non-patent literature with Office actions. Paper copies of cited references will continue to be provided by the USPTO for international applications during the international stage.

Schedule

June 2004	TCs 1600, 1700, 2800 and 2900
July 2004	TCs 3600 and 3700
August 2004	TCs 2100 and 2600

All U.S. patents and U.S. patent application publications are available on the USPTO web site. However, a simple system for downloading the cited U.S. patents and patent application publications has been established for applicants, called the E-Patent Reference system. As E-Patent Reference and Private PAIR require participating applicants to have a customer number, retrieval software and a digital certificate, all applicants are strongly encouraged to contact the Patent Electronic Business Center to acquire these items. To be ready to use this system by June 1, 2004, contact the Patent EBC as soon as possible by phone at 866-217-9197 (toll-free), 703-305-3028 or 703-308-6845 or electronically via the Internet at ebc@uspto.gov.

Other Options

The E-Patent Reference function requires the applicant to use the secure Private PAIR system, which establishes confidential communications with the applicant. Applicants using this facility must receive a digital certificate, as described above. Other options for obtaining patents which do not require the digital certificate include the USPTO's free Patents on the Web program (http://www.uspto.gov/patft/index.html). The USPTO's Office of Public Records also supplies copies of patents for a fee (http://ebiz1.uspto.gov/oems25p/index.html). Commercial sources also provide U.S. patents and patent application publications.

For complete instructions see the Official Gazette Notice, USPTO TO PROVIDE ELECTRONIC ACCESS TO CITED U.S. PATENT REFERENCES WITH OFFICE ACTIONS AND CEASE SUPPLYING PAPER COPIES, on the USPTO web site.

NOTICE OF OFFICE PLAN TO CEASE SUPPLYING COPIES OF CITED U.S. PATENT REFERENCES WITH OFFICE ACTIONS, AND PILOT TO EVALUATE THE ALTERNATIVE OF PROVIDING ELECTRONIC ACCESS TO SUCH U.S. PATENT REFERENCES

Summary

The United States Patent and Trademark Office (Office or USPTO) plans in the near future to: (1) cease mailing copies of U.S. patents and U.S. patent application publications (US patent references) with Office actions except for citations made during the international stage of an international application under the Patent Cooperation Treaty and those made during reexamination proceedings; and (2) provide electronic access to, with convenient downloading capability of, the US patent references cited in an Office action via the Office's private Patent Application Information Retrieval (PAIR) system which has a new feature called "E-Patent Reference." Before ceasing to provide copies of U.S. patent references with Office actions, the Office shall test the feasibility of the E-Patent Reference feature by conducting a two-month pilot project starting with Office actions mailed after December 1, 2003. The Office shall evaluate the pilot project and publish the results in a notice which will be posted on the Office's web site (www.USPTO.gov) and in the Patent Official Gazette (O.G.). In order to use the new E-Patent Reference feature during the pilot period, or when the Office ceases to send copies of U.S. patent references with Office actions, the applicant must: (1) obtain a digital certificate from the Office; (2) obtain a customer number from the Office, and (3) properly associate applications with the customer number. The pilot project does not involve or affect the current Office practice of supplying paper copies of foreign patent documents and non-patent literature with Office actions. Paper copies of references will continue to be provided by the USPTO for searches and written opinions prepared by the USPTO for international applications during the international stage and for reexamination proceedings.

Description of Pilot Project to Provide Electronic Access to Cited U.S. Patent References

On December 1, 2003, the Office will make available a new feature, E-Patent Reference, in the Office's private PAIR system, to allow more convenient downloading of U.S. patents and U.S. patent application publications. The new feature will allow an authorized user of private PAIR to download some or all of the U.S. patents and U.S. patent application publications cited by an examiner on form PTO-892 in Office actions, as well as U.S. patents and U.S. patent application publications submitted by applicants on form PTO/SB08 (1449) as part of an IDS. The retrieval of some or all of the documents may be performed in one downloading step with the documents encoded as Adobe Portable Document format (.pdf) files, which is an improvement over the current page-by-page retrieval capability from other USPTO systems.

Steps to Use the New E-Patent Reference Feature During the Pilot Project and Thereafter

Access to private PAIR is required to utilize E-Patent Reference. If you don't already have access to private PAIR, the Office urges practitioners, and applicants not represented by a practitioner, to take advantage of the transition period to obtain a no-cost USPTO Public Key Infrastructure (PKI) digital certificate, obtain a USPTO customer number, associate all of their pending and new application filings with their customer number, install no-cost software (supplied by the Office) required to access private PAIR and E-Patent Reference feature, and make appropriate arrangements for Internet access. The full instructions for obtaining a PKI digital certificate are available at the Office's Electronic Business Center (EBC) web page at: <http://www.uspto.gov/ebc/downloads.html>. Note that a notarized signature will be required to obtain a digital certificate.

To get a Customer Number, download and complete the Customer Number Request form, PTO-SB125, at: http://www.uspto.gov/web/forms/sb0125.pdf. The completed form can then be transmitted by facsimile to the Electronic Business Center at (703) 308-2840, or mailed to the address on the form. If you are a registered attorney or patent agent, then your registration number must be associated with your customer number. This is accomplished by adding your registration number to the Customer Number Request form. A description of associating a customer number with an application is described at the EBC web page at: http://www.uspto.gov/ebc/registration_pair.html.

The E-Patent Reference feature will be accessed using a new button on the private PAIR screen. Ordinarily all of the cited U.S. patent and U.S. patent application publication references will be available over the Internet using the Office's new E-Patent Reference feature. The size of the references to be downloaded will be displayed by E-Patent Reference so the download time can be estimated. Applicants and registered practitioners can select to download all of the references or any combination of cited references. Selected references will be downloaded as complete documents as Adobe Portable Document Format (.pdf) files. For a limited period of time, the USPTO will include a copy of this notice with Office actions to encourage applicants to use this new feature and, if needed, to take the steps outlined above in order to be able to utilize this new feature during the pilot and thereafter.

During the two-month pilot, the Office will evaluate the stability and capacity of the E-Patent Reference feature to reliably provide electronic access to cited U.S. patent and U.S. patent application publication references. While copies of U.S. patent and U.S. patent application publication references cited by examiners will continue to be mailed with Office actions during the pilot project, applicants are encouraged to use the private PAIR and the E-Patent Reference feature to electronically access and download cited U.S. patent and U.S. patent application publication references so the Office will be able to objectively evaluate its performance. The public is encouraged to submit comments to the Office on the usability and performance of the E-Patent Reference feature during the pilot. Further, during the pilot period registered practitioners, and applicants not represented by a practitioner, are encouraged to experiment with the feature, develop a proficiency in using the feature, and establish new internal processes for using the new access to the cited U.S. patents and U.S. patent application publications to prepare for the anticipated cessation of the current Office practice of supplying copies of such cited

references. The Office plans to continue to provide access to the E-Patent Reference feature during its evaluation of the pilot.

Comments

Comments concerning the E-Patent Reference feature should be in writing and directed to the Electronic Business Center (EBC) at the USPTO by electronic mail at eReference@uspto.gov or by facsimile to (703) 308-2840. Comments will be posted and made available for public inspection. To ensure that comments are considered in the evaluation of the pilot project, comments should be submitted in writing by January 15, 2004.

Comments with respect to specific applications should be sent to the Technology Centers' customer service centers. Comments concerning digital certificates, customer numbers, and associating customer numbers with applications should be sent to the Electronic Business Center (EBC) at the USPTO by facsimile at (703) 308-2840 or by e-mail at EBC@uspto.gov.

Implementation after Pilot

After the pilot, its evaluation, and publication of a subsequent notice as indicated above, the Office expects to implement its plan to cease mailing paper copies of U.S. patent references cited during examination of non provisional applications on or after February 2, 2004; although copies of cited foreign patent documents, as well as non-patent literature, will still be mailed to the applicant until such time as substantially all applications have been scanned into IFW.

For Further Information Contact

Technical information on the operation of the IFW system can be found on the USPTO website at http://www.uspto.gov/web/patents/ifw/index.html. Comments concerning the E-Patent Reference feature and questions concerning the operation of the PAIR system should be directed to the EBC at the USPTO at (866) 217-9197. The EBC may also be contacted by facsimile at (703) 308-2840 or by e-mail at EBC@uspto.gov.

Date. 12/1/03

Nicholas P. Godici

Nicholas P. Godici
Commissioner for Patents

USPTO TO PROVIDE ELECTRONIC ACCESS TO CITED U.S. PATENT REFERENCES WITH OFFICE ACTIONS AND CEASE SUPPLYING PAPER COPIES

In support of its 21st Century Strategic Plan goal of increased patent e-Government, beginning in June 2004, the United States Patent and Trademark Office (Office or USPTO) will begin the phase-in of its E-Patent Reference program and hence will: (1) **provide downloading capability of the U.S. patents and U.S. patent application publications cited in Office actions** via the E-Patent Reference feature of the Office's Patent Application Information Retrieval (PAIR) system; and (2) **cease mailing paper copies of U.S. patents and U.S. patent application publications with Office actions** (in applications and during reexamination proceedings) except for citations made during the international stage of an international application under the Patent Cooperation Treaty (PCT). In order to use the new E-Patent Reference feature applicants must: (1) obtain a digital certificate and software from the Office; (2) obtain a customer number from the Office; and (3) properly associate patent applications with the customer number. Alternatively, copies of all U.S. patents and patent application publications can be accessed without a digital certificate from the USPTO web site, from the USPTO Office of Public Records, and from commercial sources. The Office will continue the practice of supplying paper copies of foreign patent documents and non-patent literature with Office actions. Paper copies of cited references will continue to be provided by the USPTO for international applications during the international stage.

Schedule

June 2004	TCs 1600, 1700, 2800 and 2900
July 2004	TCs 3600 and 3700
August 2004	TCs 2100 and 2600

All U.S. patents and U.S. patent application publications are available on the USPTO web site. However, a simple system for downloading the cited U.S. patents and patent application publications has been established for applicants, called the E-Patent Reference system. As E-Patent Reference and Private PAIR require participating applicants to have a customer number, retrieval software and a digital certificate, all applicants are strongly encouraged to contact the Patent Electronic Business Center to acquire these items. To be ready to use this system by June 1, 2004, contact the Patent EBC as soon as possible by phone at 866-217-9197 (toll-free), 703-305-3028 or 703-308-6845 or electronically via the Internet at ebc@uspto.gov.

Other Options

The E-Patent Reference function requires the applicant to use the secure Private PAIR system, which establishes confidential communications with the applicant. Applicants using this facility must receive a digital certificate, as described above. Other options for obtaining patents which do not require the digital certificate include the USPTO's free Patents on the Web program (http://www.uspto.gov/patft/index.html). The USPTO's Office of Public Records also supplies copies of patents for a fee (http://ebiz1.uspto.gov/oems25p/index.html). Commercial sources also provide U.S. patents and patent application publications.

For complete instructions see the Official Gazette Notice, USPTO TO PROVIDE ELECTRONIC ACCESS TO CITED U.S. PATENT REFERENCES WITH OFFICE ACTIONS AND CEASE SUPPLYING PAPER COPIES, on the USPTO web site.

NOTICE OF OFFICE PLAN TO CEASE SUPPLYING COPIES OF CITED U.S. PATENT REFERENCES WITH OFFICE ACTIONS, AND PILOT TO EVALUATE THE ALTERNATIVE OF PROVIDING ELECTRONIC ACCESS TO SUCH U.S. PATENT REFERENCES

Summary

The United States Patent and Trademark Office (Office or USPTO) plans in the near future to: (1) cease mailing copies of U.S. patents and U.S. patent application publications (US patent references) with Office actions except for citations made during the international stage of an international application under the Patent Cooperation Treaty and those made during reexamination proceedings; and (2) provide electronic access to, with convenient downloading capability of, the US patent references cited in an Office action via the Office's private Patent Application Information Retrieval (PAIR) system which has a new feature called "E-Patent Reference." Before ceasing to provide copies of U.S. patent references with Office actions, the Office shall test the feasibility of the E-Patent Reference feature by conducting a two-month pilot project starting with Office actions mailed after December 1, 2003. The Office shall evaluate the pilot project and publish the results in a notice which will be posted on the Office's web site (www.USPTO.gov) and in the Patent Official Gazette (O.G.). In order to use the new E-Patent Reference feature during the pilot period, or when the Office ceases to send copies of U.S. patent references with Office actions, the applicant must: (1) obtain a digital certificate from the Office; (2) obtain a customer number from the Office, and (3) properly associate applications with the customer number. The pilot project does not involve or affect the current Office practice of supplying paper copies of foreign patent documents and non-patent literature with Office actions. Paper copies of references will continue to be provided by the USPTO for searches and written opinions prepared by the USPTO for international applications during the international stage and for reexamination proceedings.

Description of Pilot Project to Provide Electronic Access to Cited U.S. Patent References

On December 1, 2003, the Office will make available a new feature, E-Patent Reference, in the Office's private PAIR system, to allow more convenient downloading of U.S. patents and U.S. patent application publications. The new feature will allow an authorized user of private PAIR to download some or all of the U.S. patents and U.S. patent application publications cited by an examiner on form PTO-892 in Office actions, as well as U.S. patents and U.S. patent application publications submitted by applicants on form PTO/SB08 (1449) as part of an IDS. The retrieval of some or all of the documents may be performed in one downloading step with the documents encoded as Adobe Portable Document format (.pdf) files, which is an improvement over the current page-by-page retrieval capability from other USPTO systems.

Steps to Use the New E-Patent Reference Feature During the Pilot Project and Thereafter

Access to private PAIR is required to utilize E-Patent Reference. If you don't already have access to private PAIR, the Office urges practitioners, and applicants not represented by a practitioner, to take advantage of the transition period to obtain a no-cost USPTO Public Key Infrastructure (PKI) digital certificate, obtain a USPTO customer number, associate all of their pending and new application filings with their customer number, install no-cost software (supplied by the Office) required to access private PAIR and E-Patent Reference feature, and make appropriate arrangements for Internet access. The full instructions for obtaining a PKI digital certificate are available at the Office's Electronic Business Center (EBC) web page at: <http://www.uspto.gov/ebc/downloads.html>. Note that a notarized signature will be required to obtain a digital certificate.

To get a Customer Number, download and complete the Customer Number Request form, PTO-SB125, at: http://www.uspto.gov/web/forms/sb0125.pdf. The completed form can then be transmitted by facsimile to the Electronic Business Center at (703) 308-2840, or mailed to the address on the form. If you are a registered attorney or patent agent, then your registration number must be associated with your customer number. This is accomplished by adding your registration number to the Customer Number Request form. A description of associating a customer number with an application is described at the EBC web page at: http://www.uspto.gov/ebc/registration_pair.html.

The E-Patent Reference feature will be accessed using a new button on the private PAIR screen. Ordinarily all of the cited U.S. patent and U.S. patent application publication references will be available over the Internet using the Office's new E-Patent Reference feature. The size of the references to be downloaded will be displayed by E-Patent Reference so the download time can be estimated. Applicants and registered practitioners can select to download all of the references or any combination of cited references. Selected references will be downloaded as complete documents as Adobe Portable Document Format (.pdf) files. For a limited period of time, the USPTO will include a copy of this notice with Office actions to encourage applicants to use this new feature and, if needed, to take the steps outlined above in order to be able to utilize this new feature during the pilot and thereafter.

During the two-month pilot, the Office will evaluate the stability and capacity of the E-Patent Reference feature to reliably provide electronic access to cited U.S. patent and U.S. patent application publication references. While copies of U.S. patent and U.S. patent application publication references cited by examiners will continue to be mailed with Office actions during the pilot project, applicants are encouraged to use the private PAIR and the E-Patent Reference feature to electronically access and download cited U.S. patent and U.S. patent application publication references so the Office will be able to objectively evaluate its performance. The public is encouraged to submit comments to the Office on the usability and performance of the E-Patent Reference feature during the pilot. Further, during the pilot period registered practitioners, and applicants not represented by a practitioner, are encouraged to experiment with the feature, develop a proficiency in using the feature, and establish new internal processes for using the new access to the cited U.S. patents and U.S. patent application publications to prepare for the anticipated cessation of the current Office practice of supplying copies of such cited

references. The Office plans to continue to provide access to the E-Patent Reference feature during its evaluation of the pilot.

Comments

Comments concerning the E-Patent Reference feature should be in writing and directed to the Electronic Business Center (EBC) at the USPTO by electronic mail at eReference@uspto.gov or by facsimile to (703) 308-2840. Comments will be posted and made available for public inspection. To ensure that comments are considered in the evaluation of the pilot project, comments should be submitted in writing by January 15, 2004.

Comments with respect to specific applications should be sent to the Technology Centers' customer service centers. Comments concerning digital certificates, customer numbers, and associating customer numbers with applications should be sent to the Electronic Business Center (EBC) at the USPTO by facsimile at (703) 308-2840 or by e-mail at EBC@uspto.gov.

Implementation after Pilot

After the pilot, its evaluation, and publication of a subsequent notice as indicated above, the Office expects to implement its plan to cease mailing paper copies of U.S. patent references cited during examination of non provisional applications on or after February 2, 2004; although copies of cited foreign patent documents, as well as non-patent literature, will still be mailed to the applicant until such time as substantially all applications have been scanned into IFW.

For Further Information Contact

Technical information on the operation of the IFW system can be found on the USPTO website at http://www.uspto.gov/web/patents/ifw/index.html. Comments concerning the E-Patent Reference feature and questions concerning the operation of the PAIR system should be directed to the EBC at the USPTO at (866) 217-9197. The EBC may also be contacted by facsimile at (703) 308-2840 or by e-mail at EBC@uspto.gov.

Date. 12/1/03

Nicholas P. Godici

Nicholas P. Godici
Commissioner for Patents

Office Action Summary	Application No.	Applicant(s)	
	09/777,505	SHTEYN, YEVGENIY	
	Examiner	Art Unit	
	Tanim Hossain	2141	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-18 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-18 is/are rejected.

7)☒ Claim(s) 12 is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 2/05/2001 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 6.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Specification

The disclosure is objected to because of the following informalities:

a. On page 8, line 13; "of" is a typographical error which should be changed to "or".

b. On page 11, line 13; the discussion of Path 2 following steps 401, 402, etc., is incorrect, as step 404 does not logically follow the requirements of Path 2, as described in figure 4. The discussion should be corrected to read: "{401, 402, 403, 407, 411, 413}". Appropriate correction is required.

Claim Objections

Claim 12 is objected to because of the following informalities:

a. "The method of Claims 1" renders the claim ambiguous.

Appropriate correction is required.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

Claims 1-3, 6, 7, and 9-18 are rejected under 35 U.S.C. 102(a) as being anticipated by Goren-Bar (International publication WO00/08556).

As per claims 1 and 17, Goren-Bar teaches a method and a computer readable storage medium for storing instructions for carrying out a method of customizing a decision-tree based control process (page 4, lines 5-6; wherein the drawing of inferences from user information implies the use of a decision tree for the purpose of control, where certain information controls the inferences made.) that enables access to a data processing system from one of a plurality of clients (page 3, lines 21-24; where multiple computers are discussed, which implies a plurality of clients.), the method comprising: defining the decision-tree based control process, the decision tree being comprised of inter-linked decision nodes (page 3, lines 7-9; wherein the collection of "information to compute the probability of alternative user's intentions and goals or informational needs and changes the given assistance based on user competence" implies the use of inter-linked decision nodes, where distinct information collected leads to different paths in the decision nodes, which, in this case, is the given assistance.); retrieving user-information associated with a user seeking access to the system (page 4, lines 21-22); determining, from the user information, a user entry decision-node (page 5, lines 1-5; wherein the building of the user model is the entry decision-node; and configuring the control process to enable the user to access the system from the client at the user entry decision node (page 5, lines 6-14; wherein the adaptation level is the control configuration.).

As per claim 2, Goren-Bar teaches the method of claim 1, further comprising identifying the user (page 10, line 20. Also see figures 15C and D, where the user is identified by name.)

As per claim 3, Goren-Bar teaches the method of claim 1, wherein the user selects the user entry decision-node (page 11, lines 12-13; wherein if the user refuses to answer questions, default values are loaded, thus giving the user freedom to choose the entry decision-node).

As per claim 6, Goren-Bar teaches the method of claim 1 further comprising enabling the user to go from the entry decision-node to a preceding decision-node (page 12, lines 7-9; wherein a lower level in the hierarchy signifies the preceding decision-node).

As per claim 7, Goren-Bar teaches the method of claim 1, wherein the user-information is retrieved from the system (page 10, lines 20-21; wherein the user information is retrieved by supplying a user-name and password.)

As per claim 9, Goren-Bar teaches the method of claim 1, wherein the user supplies the user-information (page 4, lines 21-22).

As per claim 10, Goren-Bar teaches the method of claim 1, wherein the user-information is based on the past behavior of the user (page 5, lines 19-22).

As per claim 11, Goren-Bar teaches the method of claim 1, wherein a decision node is associated with voice prompts (page 10, line 22).

As per claim 12, Goren-Bar teaches the method of claim 1, wherein a decision node is associated with a displayed element (page 10, line 4).

As per claim 13, Goren-Bar teaches the method of claim 1, further comprising displaying a graphical user interface associated with the user entry decision-node (page 7, lines 18-22); and customizing the graphical user interface from the user-information (page 20, lines 1-10).

As per claim 14, Goren-Bar teaches the method of claim 13, further comprising enabling the user to customize the graphical user interface (page 22, lines 11-14).

As per claims 15 and 18, Goren-Bar teaches a data processing system, and a device for accessing a data processing system, comprising a client (page 7, line 26); a control process unit to execute a decision-tree based control process that enables access to the system from the client, the decision-tree being comprised of inter-linked decision nodes (figure 1, where the control process is governed by the task and user databases); a customization unit coupled to the control process unit to determine a user entry decision-node from user-information associated with a user seeking access to the system, and to configure the control process to enable the user to access the system from the client at the user entry decision-node (figure 1, where the dialog manager is the customization unit that manages the entry-points).

As per claim 16, Goren-Bar teaches the system of claim 15, further comprising a memory unit for storing the user-information (page 25, line 3; where the storing of information necessitates the existence of a memory unit).

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claim 4 is rejected under 35 U.S.C. 103(a) as being unpatentable over Goren-Bar in view of Macro Express 2000 Press Release. Goren-Bar teaches the method of claim 4, but does not specifically teach the execution of a security process suspending the customization of the control process. The press release specifically teaches executing a security process for temporarily

interrupting the customization of the control process (page 1, lines 36-42). It would have been obvious to one of ordinary skill in the art at the time of the invention to include a security feature that interrupts the customization of a control process, as taught by Macro Express 2000 in the system of Goren-Bar, as they are both from the same field of invention, namely the customization of a computer interface to make it more user-specific and efficient.

Claim 5 is rejected under 35 U.S.C. 103(a) as being unpatentable over Goren-Bar in view of Handel et al (International publication WO00/28413). Goren-Bar teaches the method of claim 4, but does not explicitly teach the dependency of the user entry decision node based on the location and physical environment of the client. Handel teaches the method of claim 1, further comprising determining the user entry decision-node on the basis of a physical environment of the client (page 37, lines 10-24). It would have been obvious to one of ordinary skill in the art at the time of the invention to include the possibility of having the user's physical environment as a basis for an entry decision node, as taught by Handel in the system of Goren-Bar, as they are both from the same field of invention, namely the customization of a computer interface to make it more convenient to the user.

Claim 8 is rejected under 35 U.S.C. 103(a) as being unpatentable over Goren-Bar in view of Kerr (European Patent Application, 0 367 709). Goren-Bar teaches the method of claim 8, but does not explicitly teach the retrieval of user-information from another party. Kerr teaches the method of claim 1, wherein the user-information is retrieved from another party (column 2, lines 49-54; where the system administrator is enabled to select a range of operations tailored specifically to the user's information to create a customized interface, which necessitates the system administrator's retrieval of the user's information). It would have been obvious to one of

ordinary skill in the art at the time of the invention to include the capability of allowing another party to retrieve the user's information, as taught by Kerr, in the system of Goren-Bar, as they are both from the same field of invention, which is the customization of a computer interface tailored to the needs of specific users.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tanim Hossain whose telephone number is 703/605-1228. The examiner can normally be reached on 8:30 am - 5 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Rupal Dharia can be reached on 703/305-4003. The fax phone number for the organization where this application or proceeding is assigned is 703/872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Tanim Hossain
Patent Examiner
Art Unit 2141

th

RUPAL DHARIA
SUPERVISORY PATENT EXAMINER

Form PTO-1449 U.S. DEPARTMENT OF COMMERCE (REV. 7-80) PATENT AND TRADEMARK OFFICE	Atty. Docket No. US 008010	Serial No.
	Applicant YEVGENIY SHTEYN	
INFORMATION DISCLOSURE CITATION (Use several sheets if necessary)	Filing Date Concurrently	Group

U.S. PATENT DOCUMENTS

Ex. Int.		Document Number	Date	Name	Class	Sub-class	Filing Date If Approp.
FB	AA	5 6 3 0 1 5 9	05/13/97	Zancho	395	800	
FB	AB	5 6 0 0 7 8 1	02/4/97	Root et al.	395	326	
FB	AC	5 4 6 5 3 5 8	11/7/95	Blades et al.	395	700	
	AD						
	AE						
	AF						

FOREIGN PATENT DOCUMENTS

		Document Number	Date	Country	Class	Sub-class	Trans. Yes	Trans. No
FB	AG	0 4 9 9 5 6 7 A 2	08/19/92				X	
FB	AH	0 3 6 7 7 0 9 A 1	05/09/90				X	
	AI							
	AJ							
	AK							

OTHER (Including Author, Title, Date, Pertinent Pages, Etc.)

	AL		~~IEEE journal of Solid-State Circuits, Richard J. Reay and Gregory T.A. Kovacs, An Unconditionally Stable Two-Stage CMOS Amplifier,~~ Date? Copy? Pages?
	AM		
	AN		

Examiner [signature] Date Considered 6/28/04

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP Draw line through citation if not in conformance and not considered. Include a copy this form with next communication to applicant.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

Application Number	09/777505
Filing Date	02/05/2001
First Named Inventor	SHTEYN, Yeggeniy E.
Art Unit	2152
Examiner Name	Unknown
Attorney Docket Number	US01 8008

U.S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number No.-Kind Code[2] (*if known*)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns Lines, Where Relevant Passages or Relevant Figures Appear
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number (ctry[3]-no.[4]-kind[5], if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of cited document	Pages, Columns Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
TH		WO 00 08556	02-17-2000	GOREN, BAR DINA		
TH		EP 0 367 709	05-09-1990	KERR, LINDA		

NON-PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in capital letters), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[6]
TH		GRAF, P: "THE SOFTWARE WORKS THE I DO", NO. 59, MAY 1999 title of item? publisher? city?	
TH		ANTON, T: "DESIGN YOUR OWN SYSTEM", NO. 59, MAY 1999	

Examiner Signature		Date Considered	6/28/04

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

[1] Unique citation designation number. [2] See attached Kinds of U.S. Patent Documents. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

Notice of References Cited	Application/Control No. 09/777,505	Applicant(s)/Patent Under Reexamination SHTEYN, YEVGENIY	
	Examiner Tanim Hossain	Art Unit 2141	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-			
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N	WO 00/08556	02-2000		Goren-Bar, Dina	
	O	WO 00/28413	05-2000		Handel, Sean P.	
	P	EP 0 367 709 A	05-1990		Kerr, Linda	
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Macro Express 2000, Press release, The Smart Way to Compute, August 24, 1999
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 9/44 — A1

(11) International Publication Number: WO 00/08556

(43) International Publication Date: 17 February 2000 (17.02.00)

(21) International Application Number: PCT/IL99/00432

(22) International Filing Date: 5 August 1999 (05.08.99)

(30) Priority Data:
125684 6 August 1998 (06.08.98) IL

(71) Applicant *(for all designated States except US)*: BEN–GURION UNIVERSITY OF THE NEGEV [IL/IL]; Research & Development Authority, P.O. Box 653, 84105 Beer–Sheva (IL).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: GOREN–BAR, Dina [IL/IL]; 6 Dror Street, P.O. Box 3048, 75130 Rishon LeZion (IL).

(74) Agents: LUZZATO, Kfir et al.; Luzzato & Luzzato, P.O. Box 5352, 84152 Beer Sheva (IL).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: METHOD FOR COMPUTER OPERATION BY AN INTELLIGENT, USER ADAPTIVE INTERFACE

(57) Abstract

Method for interactive, user adaptive operation of a computerized system by using an intelligent user interface. Information about the user and his tasks is collected and stored. A preliminary dynamic stereotype user model is built, based on predetermined default values and/or on the information about the user, as well as a task model for the user. A preliminary adaptation level of the interface is provided to the user and the user task is characterized by adaptation between the user task and the user. After a predetermined period with no user operation, assistance is offered to the user. Requests from the user are received and if found correct, executed by operating an adaptive dialog manager for the specific user. If found incorrect, instructions/help is provided to the user by the adaptive dialog manager. A user protocol representing the information about the user, collected during his operation, is generated and/or processed. Macros and/or batch automated files, representing the user's modes of operation by a sequence of operations typical for the user, are generated and/or updated. The preliminary user model, the user tasks and the user characteristics are updated in a manner responsive to the processed information from the user protocol and to successes/failures during operation of the user observed by the dialog manager. In case when a conflict between characteristics resulting from the collected information the stereotype user model occurs, the user characteristics are updated. The preliminary adaptation level of the dialog manager is modified and interaction with the user is carried out through the dialog manager according to the updated user model, user tasks and user characteristics.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

METHOD FOR COMPUTER OPERATION BY AN INTELLIGENT, USER ADAPTIVE INTERFACE

Field of the Invention

The present invention relates to the field of computer operation. More particularly, the invention relates to an improved method for user operation of computers, by using an intelligent adaptive user interface, responsive to the user operations and competence.

Background of the Invention

In the recent years, considerable efforts have been devoted to simplify the interaction between the user and the computer operated by the user. Graphical interfaces, e.g., Microsoft Windows, OSF-Motif and others have been exploited by user operating methods, based on the principle of "What You See Is What you Get" (WYSIWYG). These interfaces require less competence and knowledge from the user, but still introduced an equal base-line for any user, regardless his particular level of knowledge.

Some theories about trends of more intelligent methods using interactive interfaces between the user and the operated computer have been proposed. "Mind melding; How Far Can the Human/Computer Interface Go?" to Linderholm, Byte, Vol. 166, No. 11, 1991, p.p. 41-46 proposes computer operation by using interfaces with some degree of common-sense, multi-media, indoor large displays and user voice identification. " User-Interface Developments for the 1990's" to Marcus, Computer, Vol. 24, No. 9, 1991, p.p. 49-58 proposes computer operation by using interfaces with real time animation, means for tracing after the user eye operation and on-line error correction. "A Conversation with Don Norman" to Norman, Interactions, Vol. 2, No. 2, 1995, p.p. 47-55 even goes further by assuming computer operation by using interfaces which may be

matching the user tasks enough to eliminate the need for help during operation. Some technological efforts were devoted to such ideas, but still these efforts lack the deep understanding of the user tasks and needs.

Other operating methods provide the user tools to overcome problems which arise during the computer operation. Operating according to these "Tool Centered" methods directs the user to adjust himself to these tools, leading to a problematic mode of operation, where the user faces difficulties to interpret his wills and goals to specific and simple set of instructions to the computer. Operating methods that overcome these drawbacks should be "task oriented", i.e., adjusting their interface operation to the user needs and operating at the task level instead of the system tools level.

Nowadays, most of the widespread computer operation methods are directed to a diversity of users, each with different level of knowledge. Moreover, since modern computer systems become more and more complex, many users have only partial knowledge about the system functions and/or capabilities. In addition, different users are characterized by different needs as well as different levels of knowledge. Thus, operating methods based on a uniform interface for all users will not be sufficient.

Recently, some efforts has been devoted to try to overcome the described drawbacks. "Human-Computer Interaction" to Dix et al., Prentice-Hall, 1991 proposes an operating method using a system that collects data about the user, modeling the user, his tasks and the main subjects related to his work. This information is used, together with smart help, to support the user in a way that is most relevant to his tasks and experience. However, this method is almost not practical, since a huge amount of data, as well as large data-base is required for implementation. Furthermore,

interpretation of such data about the level of interaction between the user and the computer is very complicated.

WO 98/03907 to Horvitz et al. discloses an intelligent assistance facility helping the user during his operation. This facility comprises an event composing and monitoring system, which creates high level events from combinations of user actions by collecting visual and speech information about the user. The system uses the information to compute the probability of alternative user's intentions and goals or informational needs and changes the given assistance based on user competence. However, this user assistance facility lacks flexibility in user characterization capabilities and the ability to contest with conflicts. The system also does not consider the user's position with respect his tasks.

All the methods described above have not yet provided adequate solutions to the problem of providing an intelligent, interactive and user adaptive method for user operation, that is based on an intelligent interface, while overcome the described drawbacks. Another problematic aspect concerning these methods is how much active and/or creative should this intelligent interface be, without leading the user into confusion. Another aspect which still remains problematic, is how to contest with different and varying knowledge levels of an individual user operating a complex computerized system.

It is an object of the invention to provide a method for operating computers, while overcome the drawbacks of the prior art.

It is another object of the invention to provide a method for operating computers by using an intelligent and user friendly interface.

It is another object of the invention to provide an interface with simple and easy interaction with the user.

It is another object of the invention to provide a flexible user interface with continuous adaptation to the user.

It is another object of the invention to provide a user interface that collects information and draws inferences about the user.

It is still another object of the invention to provide a user interface which is able to handle data which is in conflict with previous data about the user.

It is yet another object of the invention to provide a flexible user interface which enables addition and modification of the user's characteristics.

Other purposes and advantages of the invention will appear as the description proceeds.

Summary of the Invention

The invention is directed to a method for interactive, user adaptive operation of a computerized system by using an intelligent user interface. Information about the user and the user tasks are collected by monitoring the user operations, and stored. Monitoring includes counting the number of times the user requested for help, the number of user errors, the time intervals between consecutive user operations and seeking after user preferences. Preferably, information about the user is collected by a questionnaire or an interview.

Preferably, a preliminary dynamic stereotype user model, based on predetermined default values and/or on the information about the user is built, as well as a task model for the user. Preferably, default values are extracted from pre-programmed assumptions, researches and studies of the addressed population of users.

A preliminary adaptation level of the interface to the user is provided. The user task is characterized by adaptation to the user, based on the collected information and the user model. Preferably, if after a predetermined period there is no user operation, assistance is offered to the user. Requests are received from the user, and executed by operating an adaptive dialog manager for the specific user, in case they are correct requests (successes). On the other hand, if the requests are incorrect (failures), instructions/help is provided by operating an adaptive dialog manager.

Preferably, information about the user, which is collected during his operation, is stored in a user protocol. User macros and/or batch automated files are generated and/or updated according to identified sequences of operations from the protocol, which are typical for the user. The preliminary user model, the user tasks and the user characteristics are updated in response to processed information from the user protocol and to successes/failures during operation of the user observed by the dialog manager. The system provides the user help in case when no task is selected for execution and corrective instructions, due to failure analysis.

In case of conflicts between characteristics resulting from the collected information and the stereotype user model, the user characteristics are updated. The preliminary adaptation is modified, and the dialog manager

interacts with the user according to the updated user model, user tasks and user characteristics.

Preferably, the user model is constructed by defining hierarchy of user stereotypes and associating characteristics for each user stereotype, wherein a value, from a predetermined scale, is assigned for each characteristic. The user preliminary model is characterized by selecting a set of stereotype attributes. The preliminary characterization is updated by modifying/adding user characteristics and/or their values based on observation. Preferably, contradictions between user characteristics are set by obtaining all the user relations to different user stereotypes and characteristics, all the user certain characteristics based on observation, and for each user characteristic with more than one value, selecting only the highest value and its associated stereotype.

Preferably, the task model is constructed by collecting and storing information about the user tasks, needs and functions and interacting with the utilities of the inherent operating system in a manner enabling execution of these utilities by the interface. Inherent utilities comprise editing, printing, reading utilities and connecting utilities to other computer networks. The inherent operating system comprises connecting utilities to other networks, such as a computer network, a web-based network, a telephone network, a cellular network, or a cable TV network.

The lowest task level is determined and each task is decomposed to a set of sub-tasks necessary to accomplish the task. Each sub-task is also decomposed iteratively, until the lowest task level is reached, and the specific sequence of tasks and/or sub-tasks is then defined. As a result, a set of individual tasks and/or jobs is output into the dialog manager.

Preferably, the user protocol is processed by counting and sorting the number of user failures and correct operations for each task, and seeking after user macros during operation and counting the frequency of each macro. The user model is updated updating the user level of knowledge, the user tasks, the user macros and the user characteristics. The user level of knowledge is updated by seeking after new information about the level of knowledge, updating or using the current level of knowledge, or using default parameters as the current level of knowledge. Each user task is updated by adding a task, in case when no task exists.

Each user macro is updated by first seeking after an existing macro. If no macro exists, the frequency of any identified sequence of user operations is counted. For any existing macro, the mean frequency per session and the general frequency of all previous sessions is calculated. A macro is generated from the identified sequence, in case when no existing macro is identified, and the frequency of the sequences is equal to or higher than a predetermined value. The mean frequency per session and the mean frequency of previous sessions is stored for each generated macro.

Preferably, interaction between the user and the dialog manager is carried out by a keyboard with suitable display, soft touch sensors, a microphone and a speaker, a Personal Digital Assistant (PDA), a cellular-phone, or a television (TV) remote-control unit, and suitable display, which may be a monitor, a soft touch display, or an interactive TV set.

The invention is also directed to a computerized system, operated by the described method. The computerized system is not limited to a specific kind, and may be any kind of a PC, a workstation, a mini-computer, a main-frame computer, a client-server system, an INTERNET server, a telemedicine network etc.

Brief Description of the Drawings

The above and other characteristics and advantages of the invention will be better understood through the following illustrative and non-imitative detailed description of preferred embodiments thereof, with reference to the appended drawings, wherein:

- Fig. 1 is a block diagram of a computerized system operated by an intelligent user interface;
- Fig. 2A is a flowchart of the operations employed by the invention for the adaptation process of the interface to the user;
- Fig. 2B is a flowchart of the operations employed by the invention for the adaptation process of the interface to the user;
- Fig. 2C is a flowchart of the operations employed by the invention for the adaptation process of the interface to the user;
- Fig. 2D is a flowchart of the operations employed by the invention for the adaptation process of the interface to the user;
- Fig. 2E is a flowchart of the operations employed by the invention for the adaptation process of the interface to the user;
- Fig. 3A is flow chart of user representation by a preliminary user model;
- Fig. 3B is flow chart of user representation by a preliminary user model;
- Fig. 4A is a flow chart of an intelligent help process according to the invention;
- Fig. 4B is a flow chart of an intelligent help process according to the invention;
- Fig. 5 is a flow chart of the process of guiding the user according to the invention;
- Fig. 6 is a flow chart of user protocol processing according to the invention;

- Fig. 7 is a flowchart of user model updating according to the invention;
- Fig. 8 is a flowchart of user macro updating according to the invention;
- Fig. 9 is a flowchart of updating of the user's level of knowledge according to the invention;
- Fig. 10 is a flowchart of updating of the tasks in the user model, according to the invention;
- Fig. 11 is a flowchart of updating of the attributes in the user model, according to the invention;
- Fig. 12 is a flowchart of an example of Hierarchical Task Analysis (HTA);
- Fig. 13 illustrates screen output displaying the main screen functions used for user modeling according to the invention;
- Fig. 14A to 14F illustrate screen outputs displaying each function from the main screen;
- Fig. 15A to 15G illustrate screen outputs displaying steps of task modeling according to the invention;
- Fig. 16A to 16C illustrate screen outputs displaying steps of adaptation of the interface to a specific user according to the invention.

Detailed Description of Preferred Embodiments

The invention provides the user of a computerized system a novel method for operating the system by interacting with an intelligent user friendly interface with adaptation to the level of competence of any specific user. Fig. 1 is a block diagram of a computerized system 10 comprising hardware and software, which is operated by the user 11 via an intelligent user interface 12. Interface 12 interacts with user 11 by employing a dialog manager 13 which collects instruction and information from the user 11 and tasks he likes to carry out, and in return offers the user 11

help and/or instructions for further operations required to accomplish the user tasks. The dialog manager 13 may interact with the user with a keyboard, soft touch sensors, microphones, speakers, an interactive television (TV) and a visual display which may comprise soft touching icons. Information about the user, which is collected in advance and/or continuously during operation, is stored in a user database 14 and is then exploited by interface 12 to build a dynamic user model which is continuously updated during operation. In addition, information about the user tasks, which is also collected in advance and/or continuously during operation, is stored in a task database 15 and is then exploited by interface 12 to build a task model. Interface 12 communicates with the computerized system 10 which executes the desired user tasks by decomposing and executing each task according to the task model. Interaction with the user is carried out by interface 12 with adaptation to the user's competence and tasks in accordance with information (about the user) extracted from the updated user model and from his task model.

According to the invention, the user is modeled by stereotype model from the collected information. A flowchart of the operations employed by the invention for the adaptation process of the interface to the user is presented in Fig. 2A. The first step 20, is identifying the user by software inputs (user name and/or a password) or by inputs provided by hardware, such as smart cards, bar-codes, sensors, voice recognition devices etc. The next step 21, is loading a preliminary user model. A flow chart of user representation by a preliminary user model is illustrated in Fig. 3A. At the first step 30 the interface checks if there is an existing model of the user. If not, the first interaction is a short interview with the user and building a model in step 31. If there is an existing model, the next step 32 is to load the model into the interface.

During a short interview with the user, questions are introduced to the user by the dialog manager 13 of Fig. 1, so as to collect preliminary information about the user and his tasks. This information comprises user personal details, occupation, position, experience with the software that executes the user tasks, and experience with similar software. In addition, several screens of the software, comprising most of the software functions are introduced and the user is asked to mark on screen the main tasks selected (by him) for execution. The user is also asked to specify the tasks in which he faced difficulties during operation and of what kinds. Information about missing functions/utilities expressed by the user is collected, as well as user preferences, e.g., how the user would like to execute a specific task. If, from any reason, the user refuses to answer to some (or all the) questions, default values or a default are loaded. These default values are extracted from previous researches and/or studies of the addressed population of potential users.

A flowchart of the interview with the user is shown in Fig. 3B. The first step 33, is seeking after the existing level of knowledge about the user, which may result from previous interview and/or previous session. At the next step 34, the user is asked to supply required (or missing) information. At the next step 35, the user response is checked. If the user refuses to answer or not responding from any reason, a default user model is generated at step 36. In case when the user cooperates, a user model is generated in step 37, according to the provided information.

A preliminary stereotype user model is built and loaded at the second step 21. A hierarchy of user stereotypes is defined to construct user classifications. The user may be associated with one or more stereotypes in any hierarchy level. For instance, a user may be an athlete and an engineer with blue eyes. Each stereotype is associated with different

characteristics where each characteristic having a weighted value from a pre-determined scale. According to the invention, this stereotype user model is able to settle contradictions between different characteristics. If there is no preliminary information about several characteristics, pre-programmed stereotype assumptions are provided based on other (known) stereotypes. For example, if the user is a software engineer, a high level of competence in computer operation is assumed. In case of a conflict between characteristics of different hierarchy levels, the characteristic having the lower level in the hierarchy will be selected. In some cases, observation on the user leads to conflicts between the observation and the taken stereotype assumption which are settled by selecting the observed characteristics. Other alternatives are determining a necessity level for each characteristic or introducing a question to the user.

Looking again at Fig 2A, the next step 22 is to load the stored information about the user task and function/position. The interface will handle differently users from different positions, even for executing the same task For example, in case when two different users, a software engineer and a secretary have the same task, like composing a letter, the interface will interact with them differently, based on the assumption that the level of knowledge of the software engineer is much higher than the level of knowledge of the secretary.

After defining the user model and loading the user tasks, the system is ready for the next step 23, in which the first adaptation to the user is implemented. A flowchart of the first adaptation is shown in Fig. 2C. At the first step 206, the first user model is built. Accordingly, in the next step 207, an interface type which matches the specific user model and user tasks is loaded. Each user model emphasizes different attributes such as

font size, density of displayed information, preferred mode of interaction (e.g., voice, editing, printing, soft touching etc.) as well as tasks. For instance, if the user age is over 60, interaction may require large icons, (relatively) few options displayed, easy communication, simple tasks, a soft touch screen etc.

In the next step 24, the interface expects the user to interact with the system. If after a predetermined period of time *T*, there is no reaction from the user the system assumes that the user is facing a difficulty and then in the next step 25 a smart (intelligent) help is offered to the user. Fig. 2B shows the content of smart help. In step 205, the system guides the user according to the current user model and level of adaptation which corresponds to the current user model. The level and kind of help is determined according to the preliminary user model. At this point, the intelligent interface starts to collect more information about the user by monitoring his operations. The reaction time of the user is measured and stored in the database and will be used later to update the user model.

A flow chart of an intelligent help process is illustrated in Fig. 4A. At the first step 40 the interface checks if there is an existing task which is selected as a goal by the user. If not, the next step 41 is to offer the user to select one. If there is a task which is a user goal, the next step 42 is to load this task.

Fig. 4B is a flow chart of task selection. At the first step 43, a list of tasks and/or macros is displayed to the user for selection. In the next step 44, the system checks if the user has selected a task. If not, the time with no task selection is counted at step 47, for a case when the user needs help. If a task is selected, the selected task, as well as the time lapsed until the selection of this task, are stored in the current session log file, at step 45.

The time lapses may indicate that some of the user tasks are his main tasks within the session, since they occupy a major portion of his time. Both time counts indications about the user preferences, as well as competence and/or experience, and used later to update the user model.

As an example for intelligent help, if the user reacts after few seconds the system classifies him as a user with high level of knowledge. On the other hand, if even after help is offered the user still does not react, the system may offer him more intensive help or even provide him instructions how to proceed.

At the next step 26 of Fig. 2A, a request for operation is received from the user and stored in a user protocol for the user reactions. This information about requests from the user is also exploited later to update the user model. For example, if the user requested an advanced function of the operated software, this may indicate on high level of knowledge. Fig. 2E is a flow chart of receiving a request from the user. In the first step 209, the system checks if any request (from the user) is received. If yes, at step 213, the request is stored in the current session log file. If no, at step 210, the system checks if the user wishes to terminate the current session. In case when the user wishes to terminate the current session, by pushing the "Esc" (escape) button, the request is set to "go to end" in step 211. In case when the user wishes to continue, the system displays instructions for the user, according to step 212.

At the next step 27, the system checks if the request from the user is correct from the aspect of the operated software. If the request is correct, the next step 29 is execution of the request. If the request is incorrect, step 28 is provides corrective instructions to the user, and going back to step 26. According to step 208, shown in Fig. 2D, the request from the user is

executed and a success is stored in the current session log file, for further adaptation.

A flow chart of guiding the user is illustrated in Fig. 5. At the first step 50 the interface checks the kind of error resulted from the user request. The next step 51 is to offer a corrective operation (solution) to the user. In the next step 52, information about the error type, solution type is stored in the log file of the current session, as well as the occurrence of the error.

In both cases, the number of successes (correct requests) and failures (incorrect requests) is stored is the user protocol, as well as the kind of corrective instructions provided to the user. This information is used to update the user model. After execution of the first request of the user, if after checking the kind of the request at step 201, the request is different than "go to end", the next request from the user is received and steps 26 to 29 of Fig. 2A are repeated iteratively, until all requested tasks are executed, where at each iteration more information about the user is collected.

At the next step 202, the interface automatically processes the user protocol to extract the required inferences about the user. A flow chart of processing of the user protocol is illustrated in Fig. 6. At the first step 60 successes as well as failures are sorted and counted . Consecutive user operations are sought at the next step 61 so as to identify potential macros. In the next step 62, identified sequences and their corresponding frequencies during the current session are stored in the session log file.

Sequences of typical user operations are sought in the next step 202 of Fig. 2A. Identified sequences are sorted and their frequency is counted . A user

macro is generated automatically in any case when the frequency of a sequence is higher than a predetermined value.

This processed information is used to update the user macros. For example, if the user interacts with a word processor, and the user has some typical preferences like having red header with bold and italic fonts comprising his name and date while typing letters, a macro that sets these kind of header is generated and operated automatically every time the user operates the word processor. This macro is updated according to the user operation at the next time he operates the same word processor.

Fig. 8 is a flowchart of updating process of the user macros. The first step 80, is to seek after an existing macro. If there is an existing macro, the mean frequency of that macro during the current session, as well as the general frequency for all past sessions, are calculated at step 82 and stored in the database. If no macro is identified, the frequency of each sequence is measured at step 81. If this frequency is less than three (or any other predetermined value) times per session, no macro is generated. If the this frequency is over than three (or any other predetermined value) times per session, a macro is generated for that sequence at step 83. The mean frequency of the new macro during the current session, as well as the general frequency for all past sessions, are calculated and stored in the database.

The final step 203, in the flowchart of Fig. 2A is updating the preliminary user model according to the information collected at the protocol during his operation. This updated user model is employed during the next interaction with the user. Fig. 7 is a flowchart of the updating process of the user model. At the first step 70 the user's level of knowledge is updated according to the processed information from the user protocol. At

the next step 71 the user tasks are updated according to the frequency of each kind of user task. If no task exists, the next task is added. The user's modes of operation stored and processed in the user protocol is updated at the next step 72. The final step 73 is updating the user characteristics in case of a conflict or when a new characteristic is disclosed after processing the user protocol.

Fig. 9 is a flowchart of updating of the user's level of knowledge. At the first step 90, the interface checks there is a new information about the level of knowledge. If not, the next step is to check if there is any level of knowledge related to the user. If not, a default value is inserted at the next step 92. If there is a new information from step 90, the next step 93 is to update the level of knowledge.

Fig. 10 is a flow chart of task updating. For every task recorded in the current session log file, the system checks, at step 110, if there are existing tasks. These tasks may be system tasks (saving, printing etc.) which are not included within the user model, or a utility in a new software (for instance, labels in Microsoft Word). If not, the frequency of each task is calculated, and the task is analyzed in step 103, looking after regular patterns which are important for starting. These patterns may be, for instance, reading E-mail in the beginning or in the end of each session, or background tasks, like looking after specific information in the Internet, on line E-mail or optimization, which continue to run in parallel with (other) current user tasks.

Fig. 11 is a flowchart of updating the user characteristics. In the first step 110, for every attribute recorded in the session log file, the system checks is there any existing attribute in the database of the user model. If no, at step 112, an attribute is added and a corresponding value is assigned to

the associated user characteristic. If yes, at the next step 111, the system checks if the new value of the characteristic equals the old value. If yes, there is no conflict. If no, this is an indication of a conflict (contradiction), between user characteristics, and conflict resolution is applied at step 113, in which the value of the characteristic with the lowing hierarchical level is selected, or values are assigned according to the level of certainty for each characteristic, or values are selected according to observations, or defining necessity level for each characteristic.

According to the invention, task modeling is required in addition to user modeling. Task modeling represents operations that should be carried out by the user to achieve his goals. A task modeling system collects inputs from three information sources: the customer, the user(s) and the designer of the computerized system. The customer (e.g., a managing director in an organization) provides inputs (e.g., answering a questionnaire) about the users, their needs, jobs, positions and their perception of using the computerized system, which are then used by the designer. The users provide inputs about their goals, preferences and needs required for functioning. The system designer provides inputs which are based on inputs from the customer and the user together with his experience in task analysis and definition.

The task modeling system provides the dialog manager two kinds of outputs: individual tasks, each comprising operations and sub-tasks that construct the task, and definition of each user position which is represented by the collection of all tasks executed by an individual user. Task analysis (or decomposition) is carried out by the system according to a pre-programmed method selected by the system designer. According to the present invention, Hierarchical Task Analysis (HTA), is used. HTA is an iterative process where each task may be decomposed to sub-tasks and

so fourth until one of a set of pre-determining basic operations is reached. HTA is easy to understand both to the user and to the system designer and may be presented graphically or verbally.

According to a preferred embodiment of the invention the specific sequence of tasks and/or sub-tasks is defined, including their attributes. These attributes may comprise the timing of carrying out the task/sub-task, a manual or a computer oriented task/sub-task, or any combination of them, and the control structure of the task/sub-task. The control structure defines if the task is carried out serially, or in parallel or iteratively, or if the execution of the task is conditioned.

An example of HTA is illustrated in Fig. 12. In this example, the task is writing a document using a word processor. The main task 120 is divided to two tasks: open an existing file 101 and begin a new file 122. Task 122 is divided to three secondary tasks: load editing screen 123, edit 124 and save 125. The save task 125 is divided to four sub-tasks: name the file 126, select drive for file saving 127, auto-save 128 and save the file in the default drive 129. In a similar way, task 121 may also be divided to sub-tasks and then to basic operations. Other (known) methods of task analysis may also be used by the present invention.

After modeling the user by the stereotype user model and the task by task analysis the dialog manager operates an adaptation process to the user model, which is derived from the user model according to his competence and level of knowledge in different relevant subjects. Several adaptation levels like maintenance, modifying defaults, monitoring the user operations, settling conflicts and updating the user model may exist. The user model is updated by modifying current values of existing characteristics and/or adding new characteristics.

For example, if the user is a 5 years old child who likes to operate a drawing software running on a PC, an initial adaptation level is determined according to the user model, based on the assumption that a 5 years old child does not read and write, is not able to operate a keyboard and may have difficulties with small details on the display. As a result, before loading the software the screen displays large icons, the background is taken from a cartoon film, instructions/help are given vocally and requests from the user are received by soft touching icons on the display. Further adaptation which is responsive to observations on the child is activated during operation.

As an illustrative example a Microsoft Windows environment was selected, comprising three demonstrations of the user modeling, the task modeling and adaptation to the user model.

Example 1 - User modeling

The user model is implemented using Microsoft Access. Implementation of the user modeling is carried out by the main screen, as shown in Fig. 13. Basically, the basic information about the user may be inserted by the system customer and the user model is built accordingly, being updated during operation. The first function in the main screen is establishment of a specific user, as shown in the screen of Fig. 14A. Basic user details like user name, user number, date of establishment and comments about the user in accordance with different categories (e.g., education and prior interaction with computers).

The second function in the main screen is relating categories to the user, as shown in the screen of Fig. 14B. The user is associated with different

stereotypes (e.g., engineers, industrial engineers, industrial engineers specialized in information systems and psychologists).

The input from the system customer may be skipped, and interaction with the user may begin (as explained before) even without any details about the user. Instead, user selected default values are loaded.

In the fourth function, different user stereotype categories are defined, as shown in the screen of Fig. 14D. Basically, these stereotype categories are constructed in hierarchical form (e.g., successors and predecessors).

After loading all the stereotypes related to the user, and/or after interaction with the user, the extracted information may cause a conflict. Different attributes may be assigned to the same characteristics by different stereotypes. The third function in the main screen, enables to overwrite attributes for each characteristic representing the user, which are used as absolute values, as shown in the screen of Fig. 14C. Different user categories are defined with associated values. In the sixth function, each user category is associated with different characteristics (e.g., associating education period and level of computer education with the category of industrial engineers) by weighted association, as shown in the screen of Fig. 14E. This weighted association is used in case of conflicts between observed data and the user model. The last function in the main screen is generating (or printing) a user report, as shown in the screen of Fig. 14F. This report is used for monitoring the user model in the interface.

Example 2 - Task modeling

Microsoft Word 6.0 (word processor) is selected for demonstrating the process of task modeling. Several modification are implemented in Word

for task definition. First, the default HNORMAL template is modified by adding "users" menu which comprises a "dialog" utility, as shown in the screen of Fig. 15A. This modification enables all previous functions of Word together with additional functions. Since each category of users carries out its typical tasks which are defined in the template, different required styles as well as special tools for each task are defined and saved as *.dot files. In addition, each template is associated with specific help files in several levels, which are normal read only Word (*.doc) files which are opened by special icons from the tool bar or alternatively by from specific menus.

After selecting "dialog" utility from "users" menu, a specific screen for selection from several options is displayed, as shown in the screen of Fig. 15B. These options are related to tasks of an un-experienced user, a secretary and students. Other options like a screen with Qtext word processor (QTX) format, general purpose screen and article typing screen are available. Tool bars are adapted to the task according to collected information. For instance, a tool bar containing only the basic functions for editing and printing is displayed to an un-experienced user, as shown in the screen of Fig. 15C. Other users experienced in QTX who face difficulties with icon size may use a "QTX compatible" screen, shown in the Fig. 15D. Another screen, shown in Fig. 15E, is dedicated for preparing an academic article. This screen enables typing in two columns as well as inserting tables and graphical objects into the text.

The screen shown in Fig. 15 F contains several tasks which are typical to a secretary (e.g., financial transfers, typing a memorandum, typing a fax cover sheet and typing a meeting protocol). Selecting a financial transfer option, for instance, leads to a dedicated screen for that task, as shown in Fig. 15G. All dedicated (selectable) formats are prepared in advance

according to previous standards. There is also a possibility that the user creates a form and adds it to the screen for future use.

Example 3 - Adaptation of the dialog level

Adaptation to the user is expressed in this example by forming the screen as well as the format and content of the help program. After selecting the "startmenu" box, a screen which defines the level of user is displayed, as shown in Fig. 16A. The user may select the "novice" box or the "advanced" box. If "novice" box is selected, instead of a standard (and complicated for "novice") Word toolbar, a screen with help toolbar comprising six help boxes (Scope, Applicable Documents, Engineering Requirements, Qualification Requirements, Preparation for Delivery and Notes) about different subject is displayed, as shown in Fig. 16B. These six subjects represent the subjects used for composing a Software Requirement Specifications (SRS) document involved in software development projects. By selecting the "Scope" box, for instance, an extended help screen is displayed to a user with no experience in preparing SRS documents, as shown in Fig. 16C.

Another advanced help subjects are introduced to an advanced user by selecting the "advanced" box. When an advanced user is operating the software, the advanced help box disappears after the third time help is requested during the same session. In this manner, a dynamic adaptation to the user level of knowledge is implemented, based on the assumption that after a determined number (three in this case) of times a direct access to advanced help is no more necessary. Of course, if the user likes to continue with the advanced help, he may select the subject box again and have the same direct access to advanced help for three more times during the session.

Of course, the above examples and description has been provided only for the purpose of illustrations, and are not intended to limit the invention in any way. The present invention is not restricted to Windows environment, and may be carried out in different environment of different data bases, such as relational, object oriented and others. As will be appreciated by the skilled person, the invention can be carried out in a great variety of ways, employing more than one technique from those described above, all without exceeding the scope of the invention.

CLAIMS

1. Method for interactive, user adaptive operation of a computerized system by using an intelligent user interface, comprising the steps of:

a) collecting and storing information about the user;

b) collecting and storing information about the user task;

c) building a preliminary dynamic stereotype user model based on predetermined default values and/or on the information about the user;

d) building a task model for the user;

e) determining and providing a preliminary adaptation level of the interface to the user;

f) characterizing the user task by adaptation between the user task and the user;

g) offering the user assistance after a predetermined period with no user operation;

h) receiving requests from the user and executing them by operating an adaptive dialog manager for the specific user, in case of correct requests indicating the kind and number of user successes;

i) receiving a request from the user and providing the user instructions/help by operating an adaptive dialog manager for the specific user, in case of incorrect requests indicating the kind and number of user failures;

j) generating and/or processing a user protocol representing the information about the user collected during his operation;

k) generating and/or updating macros and/or batch automated files representing the user's modes of operation by a sequence of operations typical for the user;

l) updating the preliminary user model, the user tasks and the user characteristics in a manner responsive to the processed information from the user protocol and to successes/failures during operation of the user observed by the dialog manager;

m) updating the user characteristics in case of occurrence of a conflict between characteristics resulting from the collected information the stereotype user model;

n) modifying the preliminary adaptation level of the dialog manager; and

o) interacting with the user through the dialog manager according to the updated user model, user tasks and user characteristics.

2. Method according to claim 1, wherein information about the user preferences is collected by monitoring the user operations;

3. Method according to claim 2, wherein the number of times the user requested for help being counted.

4. Method according to claim 2, wherein the number of user errors during operation being counted and interpreted.

5. Method according to claim 2, wherein time intervals between consecutive user operations being measured.

6. Method according to claim 2, wherein the user preferences are monitored during operation.

7. Method according to claim 1, wherein information about the user is collected by first introducing a questionnaire to the user.

8. Method according to claim 1, wherein information about the user is collected from a preliminary interview with the user.

9. Method according to claim 1, wherein default values are extracted from pre-programmed assumptions.

10. Method according to claim 1, wherein default values are extracted from researches on the addressed population of users.

11. Method according to claim 1, wherein default values are extracted from studies of the addressed population of users.

12. Method according to claim 1, wherein the user model is constructed by the steps of:

a) defining hierarchy of user stereotypes representing different user classifications;

b) associating objective and/or subjective characteristics for each user stereotype;

c) assigning a value for each characteristic;

d) representing the connection between the user classification and the user stereotype by a corresponding value from a predetermined scale;

e) characterizing the user preliminary model by selecting a set of stereotype attributes; and

f) updating the preliminary characterization by modifying/adding user characteristics and/or their values based on observation.

13. Method according to claim 12, wherein the user is further characterized by settling contradictions between user characteristics by the steps of:

a) obtaining all the user direct/indirect relations to different user stereotypes;

b) obtaining all the user direct/indirect relations to different characteristics existing in the stereotypes to which the user being related;

c) obtaining all the user certain characteristics based on observation; and

d) for each user characteristic with more than one value, selecting only the highest value and its associated stereotype.

14. Method according to claim 1, wherein the task model is constructed by the steps of:

a) collecting and storing information about the user tasks from the customer, the user and the system designer;

b) collecting and storing information about the user needs from the customer, the user and the system designer;

c) collecting and storing information about the user functions from the customer, the user and the system designer;

d) interacting with the utilities of the inherent operating system in a manner enabling execution of these utilities by the interface;

e) determining the lowest task level;

f) decomposing each task to a set of sub-tasks necessary to accomplish the task;

g) decomposing each sub-task iteratively, until the lowest task level is reached;

h) defining the specific sequence of tasks and/or sub-tasks; and

i) outputting a set of individual tasks and/or jobs representing several tasks, into the dialog manager.

15. Method according to claim 14, further comprising determining the attributes of the tasks and/or sub-tasks.

16. Method according to claim 15, wherein the attributes of the tasks and/or sub-tasks are selected from the following group of attributes:

- the timing of carrying out the task/sub-task;
- a manual or a computer oriented task/sub-task, or any combination thereof; and
- the control structure of said task/sub-task.

17. Method according to claim 15, wherein the control structure of the tasks and/or sub-tasks is selected from the following group of control structures:

- a serial structure;
- a parallel structure;
- an iterative structure; and
- a conditioned structure.

18. Method according to claim 14, wherein the inherent operating system comprises editing utilities.

19. Method according to claim 14, wherein the inherent operating system comprises printing utilities.

20. Method according to claim 14, wherein the inherent operating system comprises reading utilities.

21. Method according to claim 14, wherein the inherent operating system comprises connecting utilities to other networks.

22. Method according to claim 21, wherein each other network is selected from the following group of networks:

- a computer network;

- a web-based network;
- a telephone network;
- a cellular network; and
- a cable TV network.

23. Method according to claim 1, wherein the system provides the user help in case when no task is selected for execution.

24. Method according to claim 1, wherein the system analyzes the type of failure during the user operation and provides the user corrective instructions.

25. Method according to claim 1, wherein the user protocol is processed by the steps of:

a) for each task, counting and sorting the number of user correct operations;

b) for each task, counting and sorting the number of user failures; and

c) seeking after user macros during operation and counting the frequency of each macro.

26. Method according to claim 1, wherein the user model is updated by the steps of:

a) updating the user level of knowledge;

b) updating the user tasks;

c) updating the user macros; and

d) updating the user characteristics.

27. Method according to claim 21, wherein the user level of knowledge is updated by the steps of:

a) seeking after new information about the level of knowledge;

b) updating the current level of knowledge in case when new information is identified;

c) using the current level of knowledge in case when no new information is identified and a level of knowledge exists; or

d) using default parameters as the current level of knowledge in case when no new information is identified and no level of knowledge exists.

28. Method according to claim 26, wherein each user task is updated by adding a task in case when no task exists.

29. Method according to claim 26, wherein each user macro is updated by the steps of:

a) seeking after an existing macro;

b) calculating the mean frequency per session and the general frequency of all previous sessions, in case when an existing macro is identified;

c) counting the frequency of any identified sequence of user operations, in case when no existing macro is identified;

d) generating a macro from the identified sequence, in case when no existing macro is identified and the frequency of step c) above is equal to or higher than a predetermined value; and

e) for each generated macro, storing the mean frequency per session and the mean frequency of previous sessions.

30. Method according to claim 26, wherein each user characteristic is updated by settling contradictions between user characteristics

31. Method according to claim 1, wherein interaction between the user and the dialog manager is carried out by a keyboard with suitable display.

32. Method according to claim 1, wherein interaction between the user and the dialog manager is carried out by soft touch sensors with suitable display.

33. Method according to claim 1, wherein interaction between the user and the dialog manager is carried out by a microphone and a speaker with suitable display.

34. Method according to claim 1, wherein interaction between the user and the dialog manager is carried out by a suitable means, selected from the following group:

- a soft touch display;
- PDA;
- TV remote control unit;
- cellular telephone; and
- interactive TV.

35. Method according to any one of claims 1 to 34, wherein the computerized system is a PC.

36. Method according to any one of claims 1 to 34, wherein the computerized system is a workstation.

37. Method according to any one of claims 1 to 34, wherein the computerized system is a mini-computer.

38. Method according to any one of claims 1 to 34, wherein the computerized system is a main-frame computer.

39. Method according to any one of claims 1 to 34, wherein the computerized system is a client-server system.

40. Method according to any one of claims 1 to 34, wherein the computerized system is an INTERNET server.

41. Method according to any one of claims 1 to 34, wherein the computerized system is a telemedicine network.

42. A computerized system comprising:

a) computer hardware and peripheral devices;

b) an operating system software for running user applications;

c) a user application software running by the operating system; and

d) an intelligent user adaptive interface for interaction between the user and the operating system and/or application software.

43. An intelligent user adaptive interface according to claim 42, comprising:

a) means for collecting and storing information about the user;

b) means for collecting and storing information about the user task;

c) means for building a preliminary dynamic stereotype user model based on predetermined default values and/or on the information about the user;

d) means for building a task model for the user;

e) means for determining and providing a preliminary adaptation level of the interface to the user;

f) means for characterizing the user task by adaptation between the user task and the user;

g) means for offering the user assistance;

h) means for receiving requests from the user and executing them by operating an adaptive dialog manager for the specific user;

i) means for generating and/or processing a user protocol representing the information about the user collected during his operation;

j) means for generating and/or updating macros and/or batch automated files representing the user's modes of operation;

k) means for updating the preliminary user model, the user tasks and the user characteristics in a manner responsive to the processed information from the user protocol and to successes/failures during operation of the user observed by the dialog manager;

l) means for updating the user characteristics in case of occurrence of a conflict between characteristics resulting from the collected information the stereotype user model;

m) means for modifying the preliminary adaptation level of the dialog manager; and

n) means for interacting with the user trough the dialog manager according to the updated user model, user tasks and user characteristics.

44. An intelligent user adaptive interface according to claim 43, comprising means for monitoring the user operations.

45. An intelligent user adaptive interface according to claim 43, comprising means for counting and interpreting the number of times the user requested for help.

46. An intelligent user adaptive interface according to claim 43, comprising means for counting and interpreting the number of errors during user operation.

47. An intelligent user adaptive interface according to claim 43, comprising means for measuring the time intervals between consecutive user operations.

48. An intelligent user adaptive interface according to claim 43, comprising means for monitoring the user preferences during operation.

49. An intelligent user adaptive interface according to claim 43, comprising means for providing the user help in case when no task is selected for execution.

50. An intelligent user adaptive interface according to claim 43, comprising means for analyzing the type of failure during the user operation and means for providing the user corrective instructions.

51. An intelligent user adaptive interface according to claim 43, where interaction with the user is carried out by a keyboard with suitable display.

52. An intelligent user adaptive interface according to claim 43, where interaction with the user is carried out by soft touch sensors with suitable display.

53. An intelligent user adaptive interface according to claim 43, where interaction with the user is carried out by a microphone and a speaker with suitable display.

54. An intelligent user adaptive interface according to claim 43, where interaction with the user is carried out by a suitable soft touch display.

55. An intelligent user adaptive interface according to claim 54, in which interaction between the user and the dialog manager is carried out by a suitable means, selected from the following group:

- a soft touch display;
- a TV set;
- PDA;
- TV remote control unit;
- cellular telephone; and
- interactive TV.

56. Method for interactive, user adaptive operation of a computerized system by using an intelligent user interface, substantially as described and illustrated.

57. A computerized system, operated by interacting with an intelligent, user adaptive interface, substantially as described and illustrated.

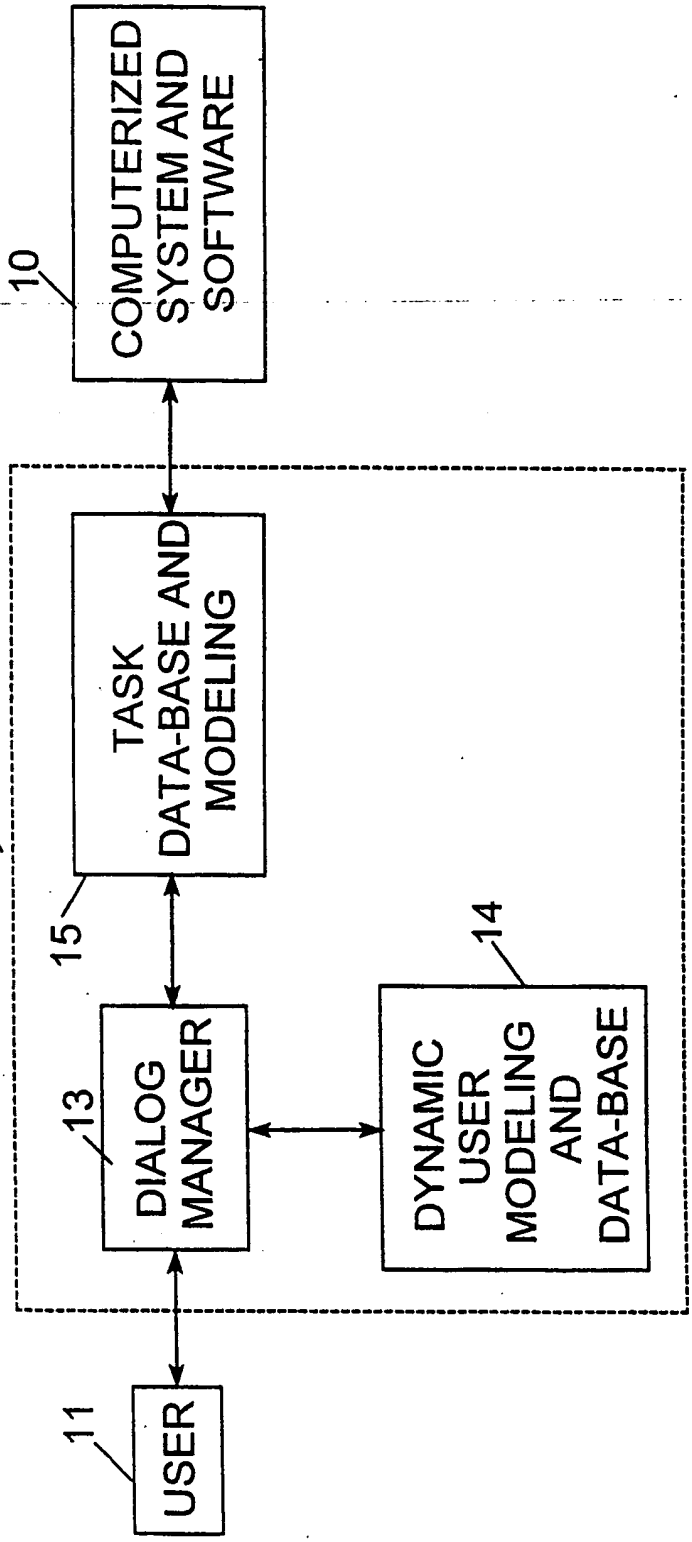

COMPUTERIZED SYSTEM AND SOFTWARE
10
TASK DATA-BASE AND MODELING
12
15
DIALOG MANAGER
13
DYNAMIC USER MODELING AND DATA-BASE
14
USER
11

Fig. 1

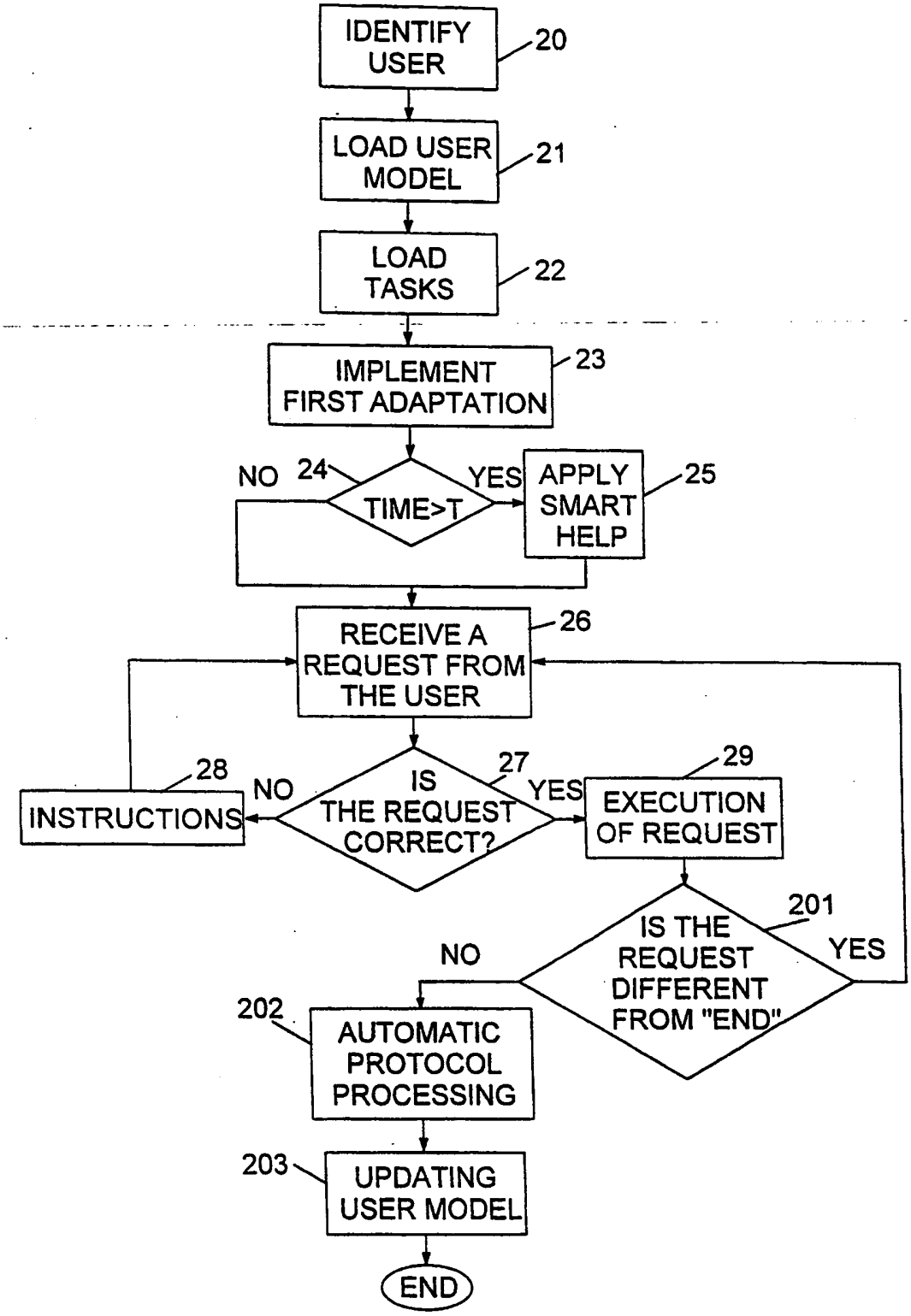

IDENTIFY USER
20
LOAD USER MODEL
21
LOAD TASKS
22
IMPLEMENT FIRST ADAPTATION
23
NO
24
TIME>T
YES
APPLY SMART HELP
25
RECEIVE A REQUEST FROM THE USER
26
28
INSTRUCTIONS
NO
27
IS THE REQUEST CORRECT?
YES
29
EXECUTION OF REQUEST
201
IS THE REQUEST DIFFERENT FROM "END"
NO
YES
202
AUTOMATIC PROTOCOL PROCESSING
203
UPDATING USER MODEL
END

Fig. 2A

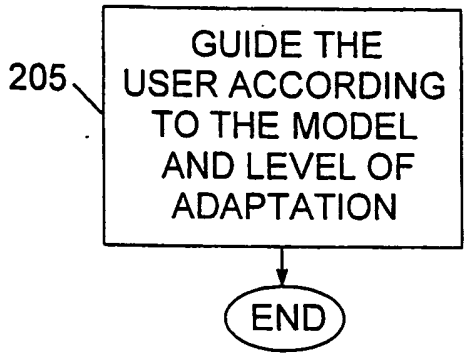


Fig. 2B

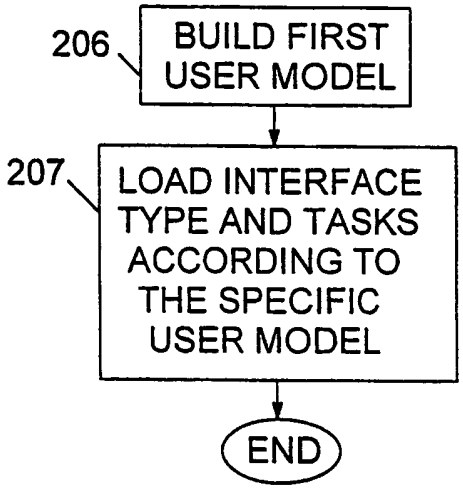


Fig. 2C

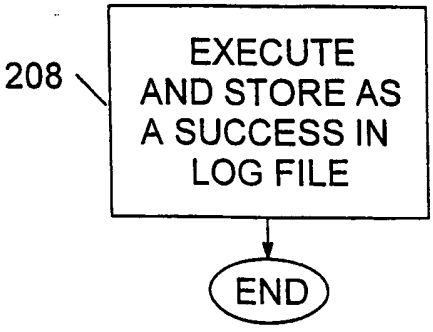


Fig. 2D

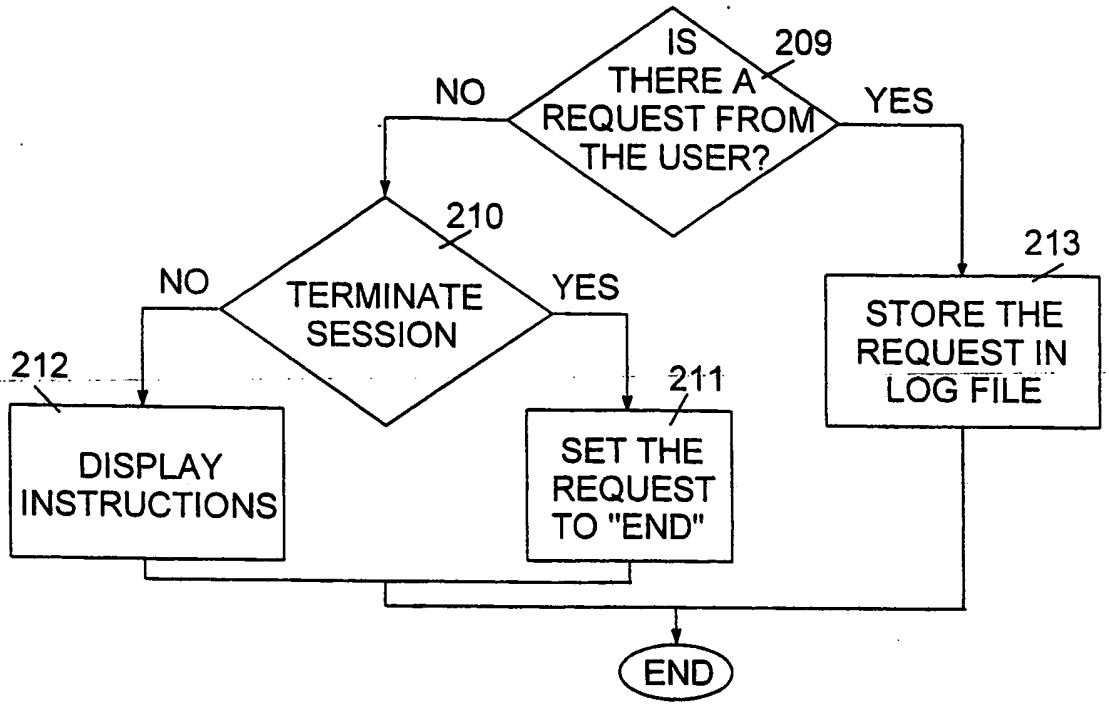

IS THERE A REQUEST FROM THE USER?
209
NO
YES
TERMINATE SESSION
210
NO
YES
212
211
213
DISPLAY INSTRUCTIONS
SET THE REQUEST TO "END"
STORE THE REQUEST IN LOG FILE
END

Fig. 2E

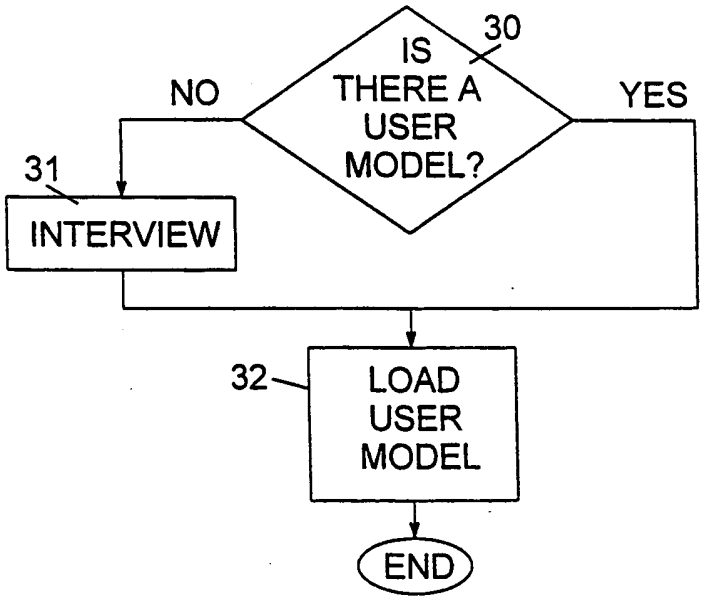

IS THERE A USER MODEL?
30
NO
YES
31
INTERVIEW
32
LOAD USER MODEL
END

Fig. 3A

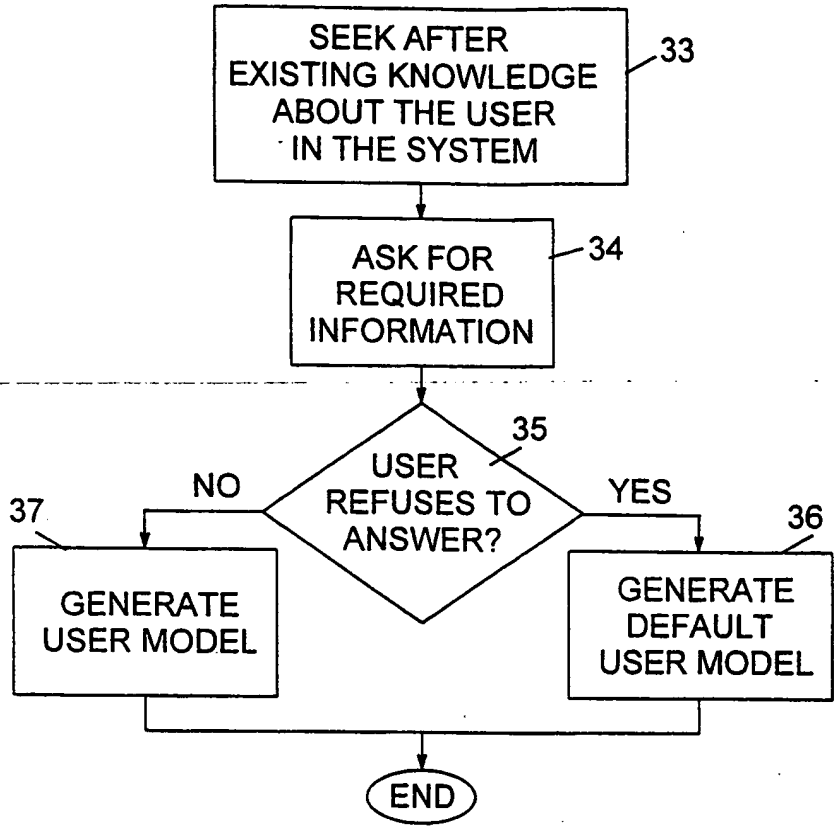

SEEK AFTER EXISTING KNOWLEDGE ABOUT THE USER IN THE SYSTEM
33
ASK FOR REQUIRED INFORMATION
34
USER REFUSES TO ANSWER?
35
NO
YES
37
36
GENERATE USER MODEL
GENERATE DEFAULT USER MODEL
END

Fig. 3B

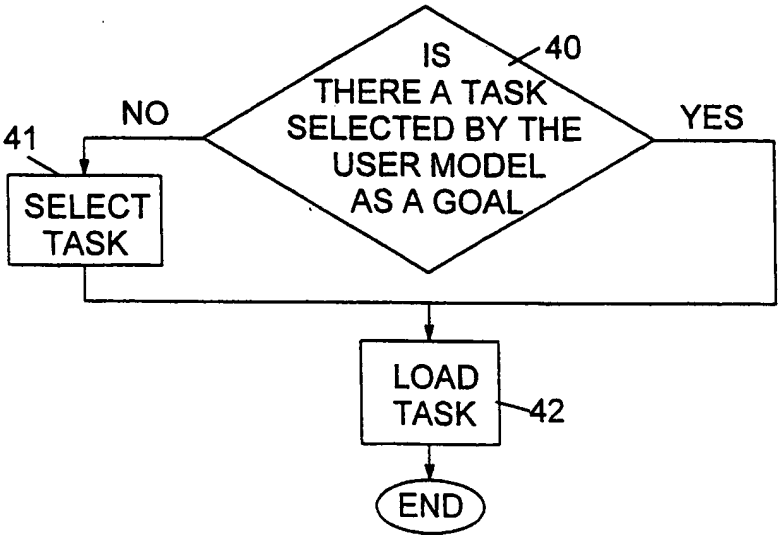

IS THERE A TASK SELECTED BY THE USER MODEL AS A GOAL
40
NO
YES
41
SELECT TASK
LOAD TASK
42
END

Fig. 4A

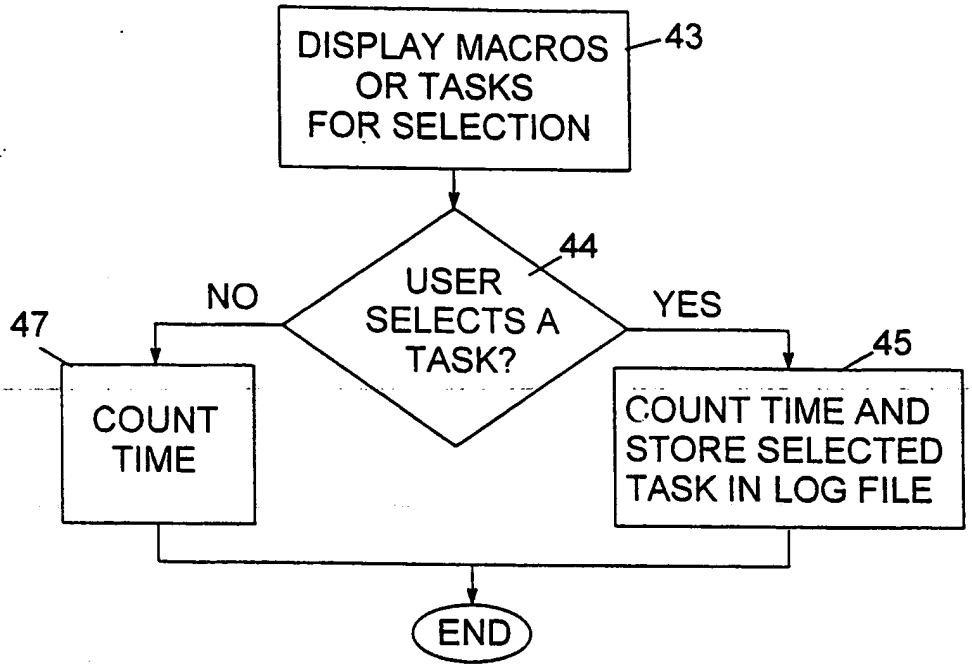

DISPLAY MACROS OR TASKS FOR SELECTION
43
USER SELECTS A TASK?
44
NO
YES
47
45
COUNT TIME
COUNT TIME AND STORE SELECTED TASK IN LOG FILE
END

Fig. 4B

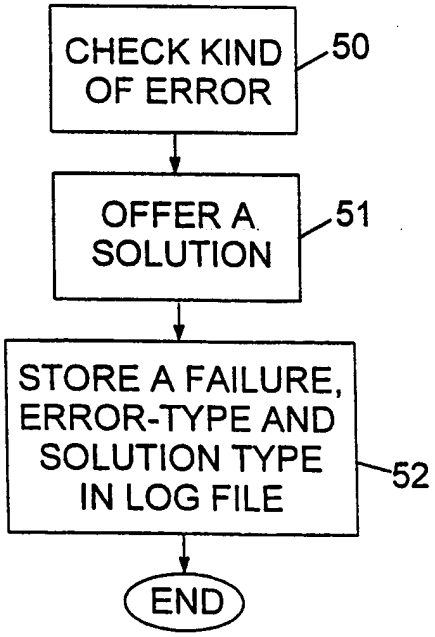

CHECK KIND OF ERROR
50
OFFER A SOLUTION
51
STORE A FAILURE, ERROR-TYPE AND SOLUTION TYPE IN LOG FILE
52
END

Fig. 5

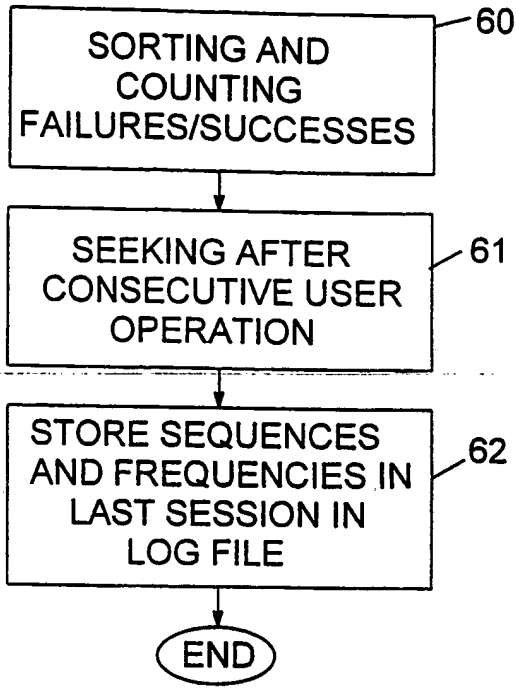

SORTING AND COUNTING FAILURES/SUCCESSES
60
SEEKING AFTER CONSECUTIVE USER OPERATION
61
STORE SEQUENCES AND FREQUENCIES IN LAST SESSION IN LOG FILE
62
END

Fig. 6

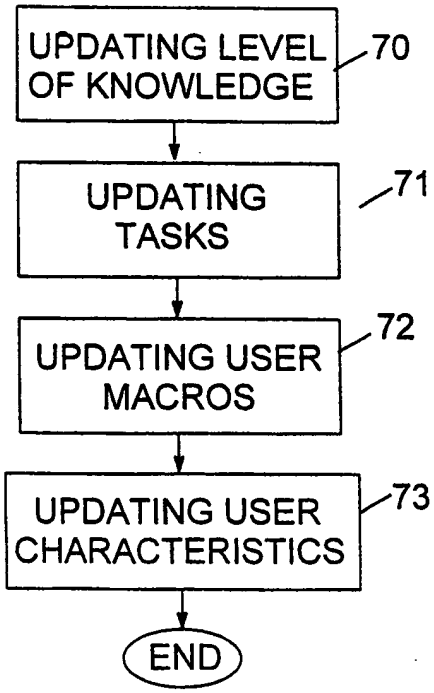

UPDATING LEVEL OF KNOWLEDGE
70
UPDATING TASKS
71
UPDATING USER MACROS
72
UPDATING USER CHARACTERISTICS
73
END

Fig. 7

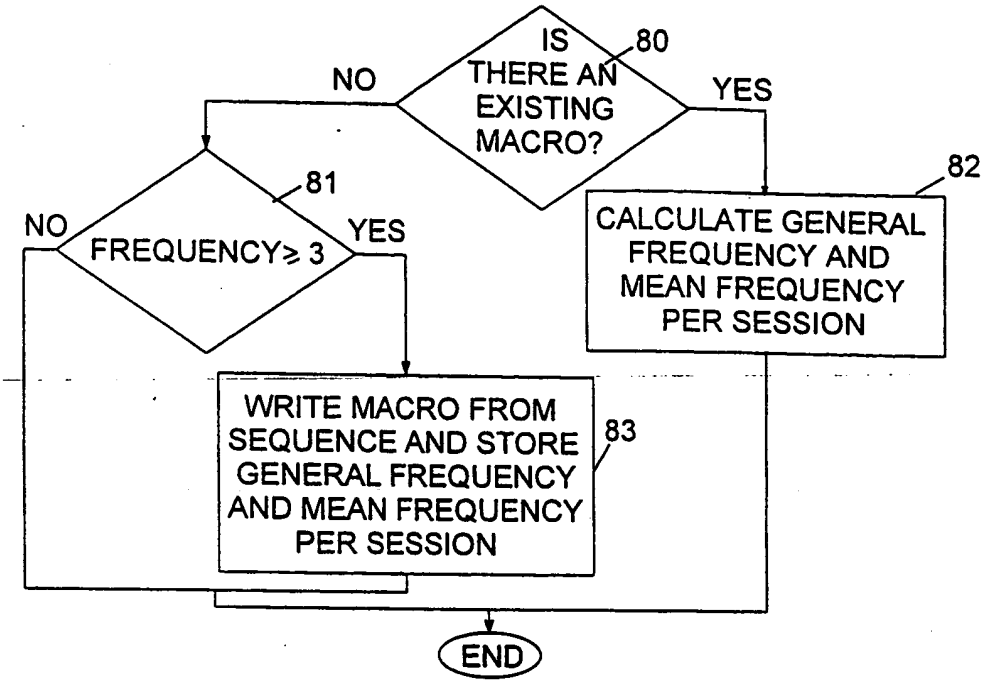

IS THERE AN EXISTING MACRO?
80
NO
YES
82
CALCULATE GENERAL FREQUENCY AND MEAN FREQUENCY PER SESSION
81
NO
FREQUENCY≥ 3
YES
WRITE MACRO FROM SEQUENCE AND STORE GENERAL FREQUENCY AND MEAN FREQUENCY PER SESSION
83
END

Fig. 8

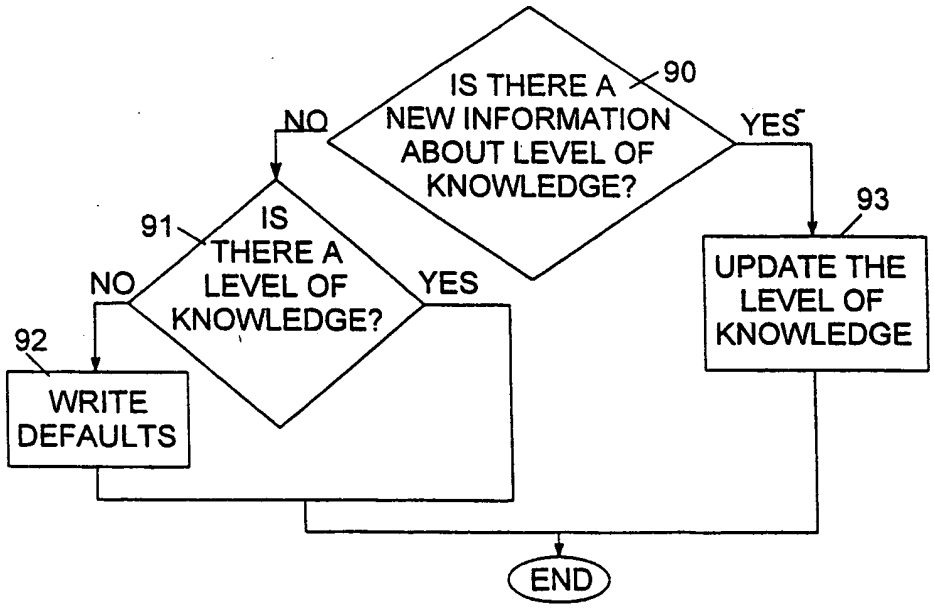

IS THERE A NEW INFORMATION ABOUT LEVEL OF KNOWLEDGE?
90
NO
YES
93
UPDATE THE LEVEL OF KNOWLEDGE
91
IS THERE A LEVEL OF KNOWLEDGE?
NO
YES
92
WRITE DEFAULTS
END

Fig. 9

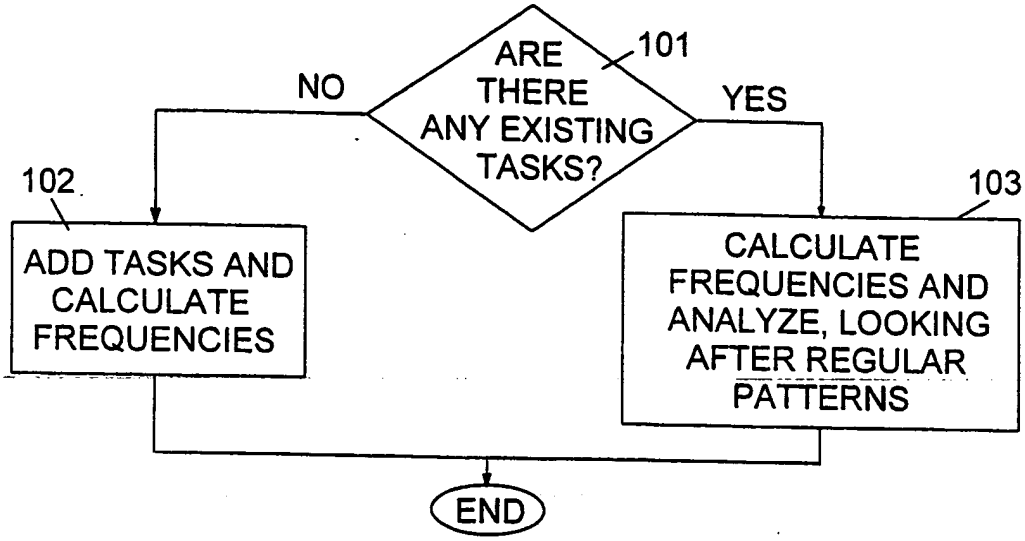

ARE THERE ANY EXISTING TASKS?
101
NO
YES
102
ADD TASKS AND CALCULATE FREQUENCIES
103
CALCULATE FREQUENCIES AND ANALYZE, LOOKING AFTER REGULAR PATTERNS
END

Fig. 10

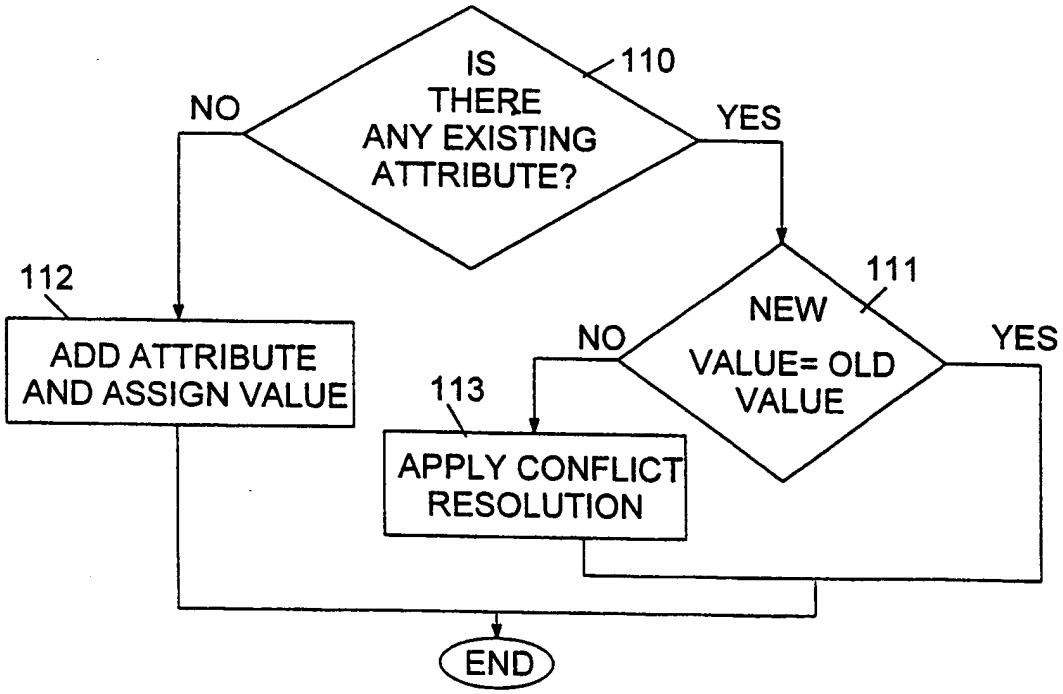

IS THERE ANY EXISTING ATTRIBUTE?
110
NO
YES
112
ADD ATTRIBUTE AND ASSIGN VALUE
111
NEW VALUE= OLD VALUE
NO
YES
113
APPLY CONFLICT RESOLUTION
END

Fig. 11

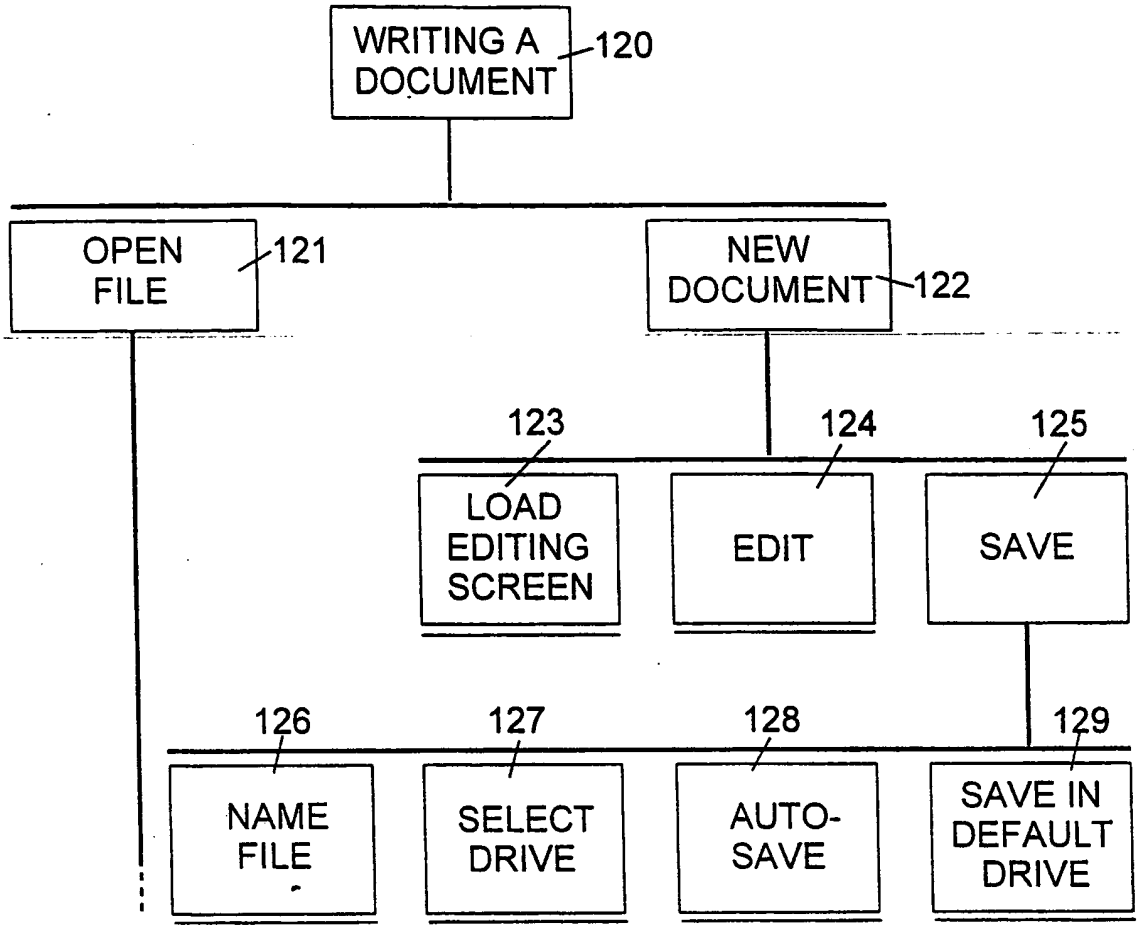

WRITING A DOCUMENT
120
OPEN FILE
121
NEW DOCUMENT
122
123
124
125
LOAD EDITING SCREEN
EDIT
SAVE
126
127
128
129
NAME FILE
SELECT DRIVE
AUTO-SAVE
SAVE IN DEFAULT DRIVE

Fig. 12

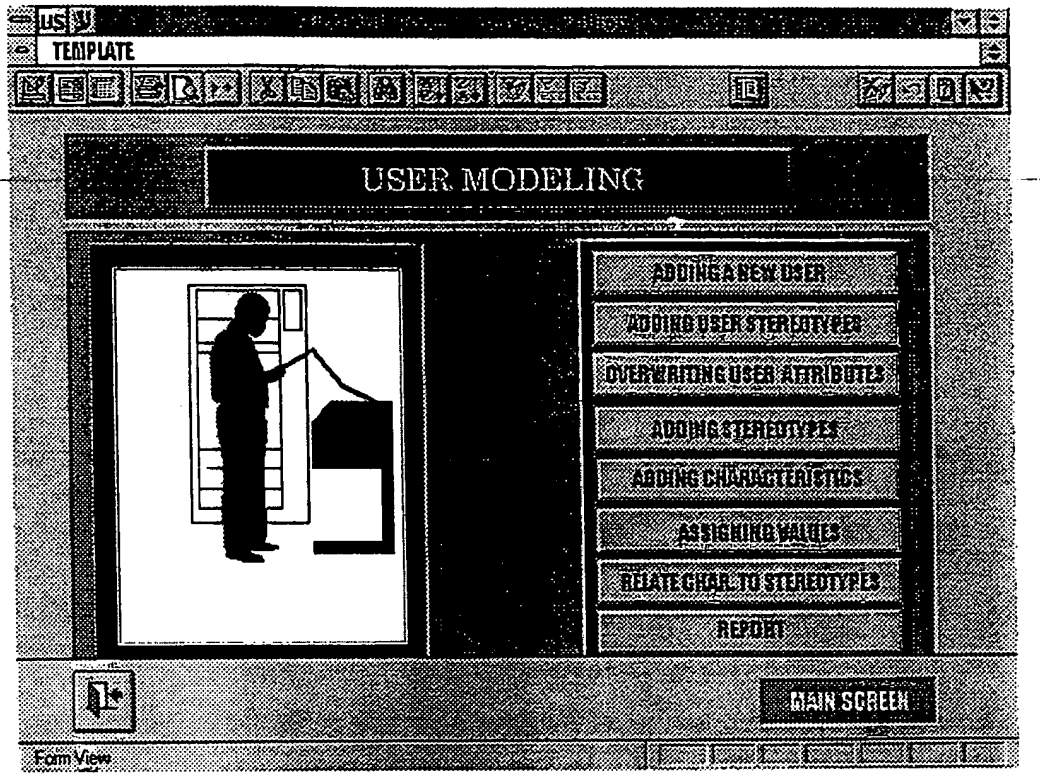

TEMPLATE
USER MODELING
ADDING A NEW USER
ADDING USER STEREOTYPES
OVERWRITING USER ATTRIBUTES
ADDING STEREOTYPES
ADDING CHARACTERISTICS
ASSIGNING VALUES
RELATE CHAR. TO STEREOTYPES
REPORT
MAIN SCREEN
Form View

Fig. 13

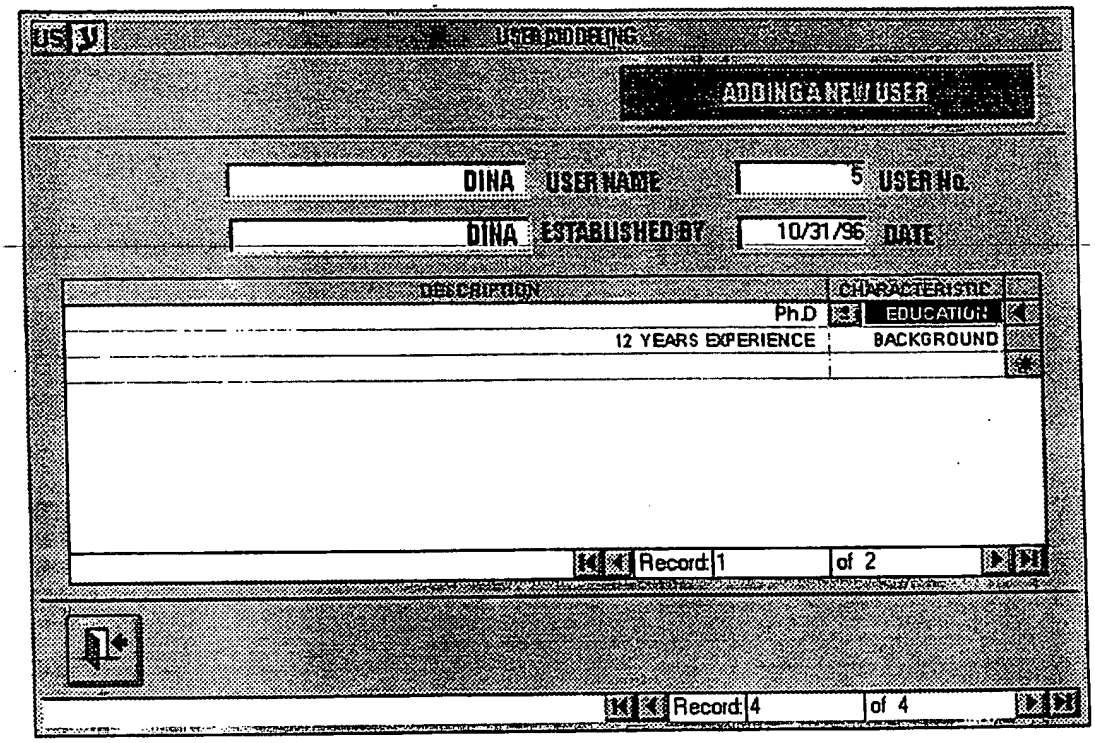

USER MODELING
ADDING A NEW USER
DINA
USER NAME
5
USER No.
DINA
ESTABLISHED BY
10/31/96
DATE
DESCRIPTION
CHARACTERISTIC
Ph.D
EDUCATION
12 YEARS EXPERIENCE
BACKGROUND
Record: 1
of 2
Record: 4
of 4

Fig. 14A

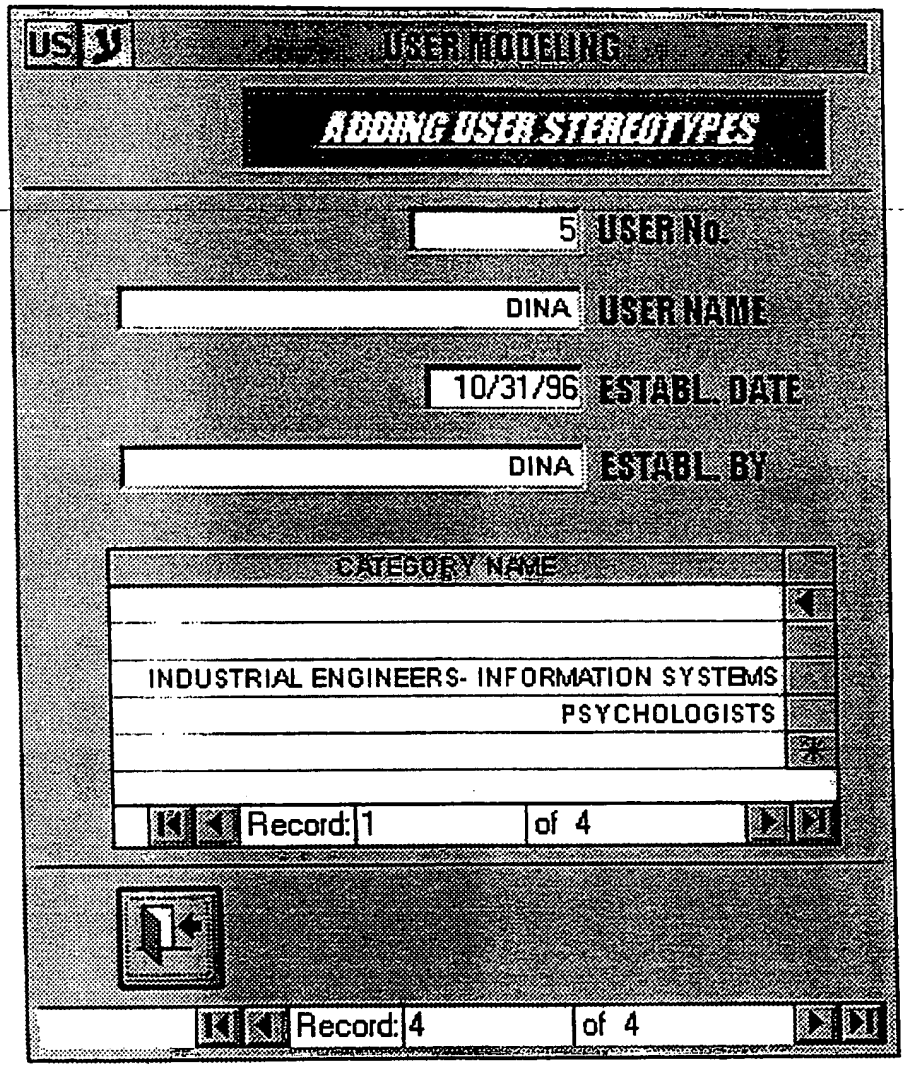

USER MODELING
ADDING USER STEREOTYPES
5
USER No.
DINA
USER NAME
10/31/96
ESTABL. DATE
DINA
ESTABL. BY
CATEGORY NAME
INDUSTRIAL ENGINEERS- INFORMATION SYSTEMS
PSYCHOLOGISTS
Record: 1 of 4
Record: 4 of 4

Fig. 14B

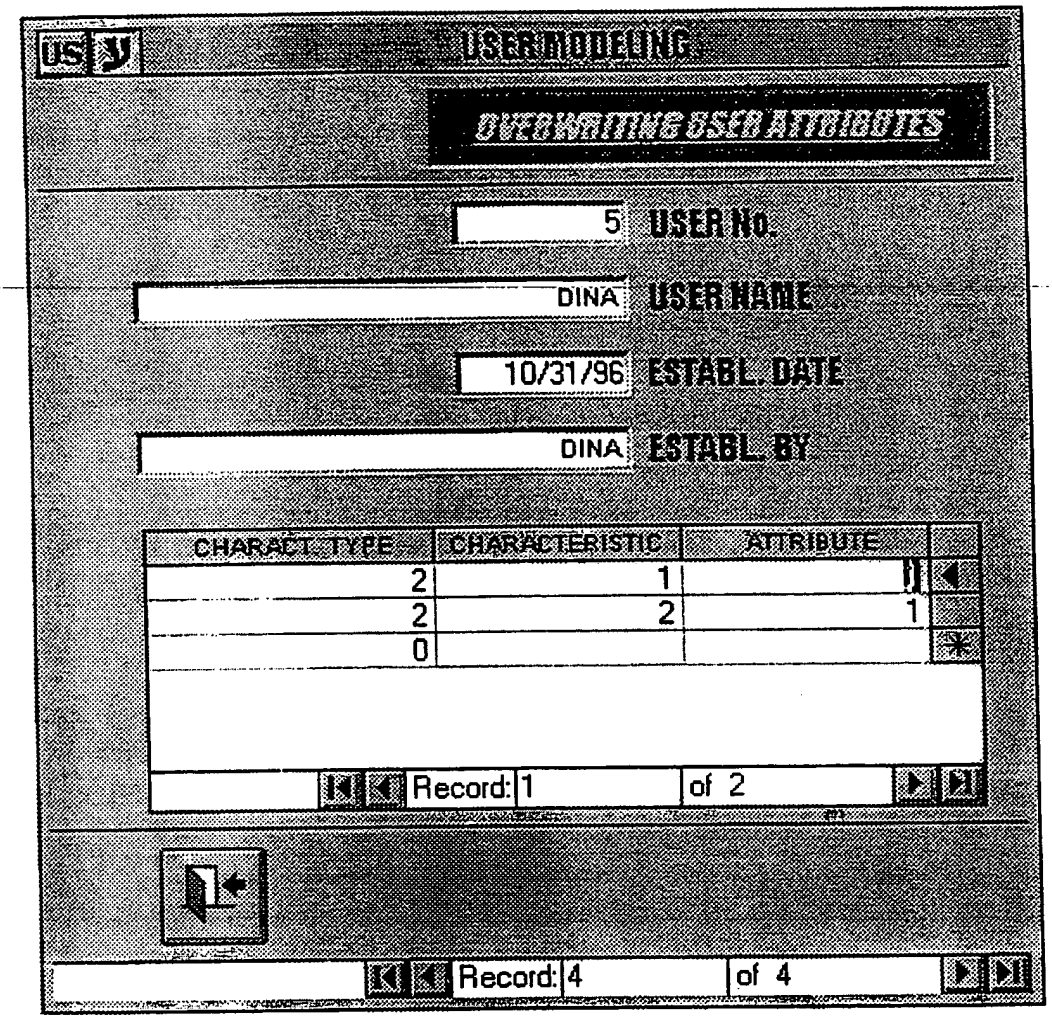

USER MODELING
5
USER No.
DINA
USER NAME
10/31/96
ESTABL. DATE
DINA
ESTABL. BY
CHARACT. TYPE
CHARACTERISTIC
ATTRIBUTE
2 1
2 2 1
0
Record: 1 of 2
Record: 4 of 4

Fig. 14C

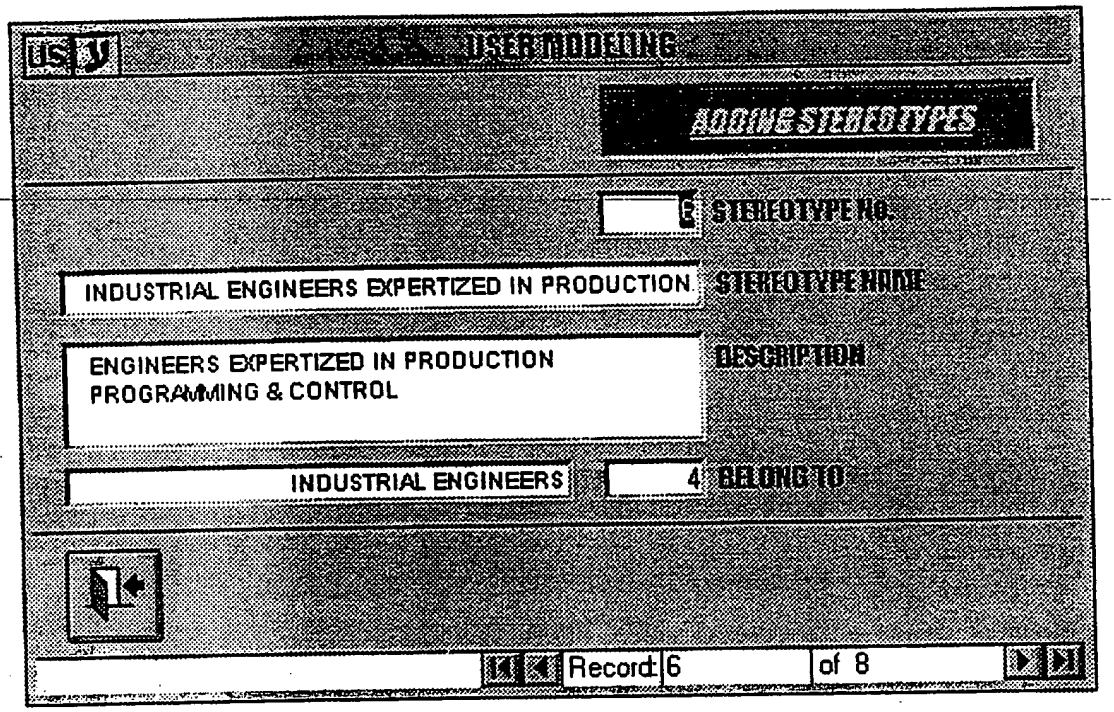

USER MODELING
ADDING STEREOTYPES
6
STEREOTYPE No.
INDUSTRIAL ENGINEERS EXPERTIZED IN PRODUCTION
STEREOTYPE NAME
ENGINEERS EXPERTIZED IN PRODUCTION
PROGRAMMING & CONTROL
DESCRIPTION
INDUSTRIAL ENGINEERS
4
BELONG TO
Record: 6
of 8

Fig. 14D

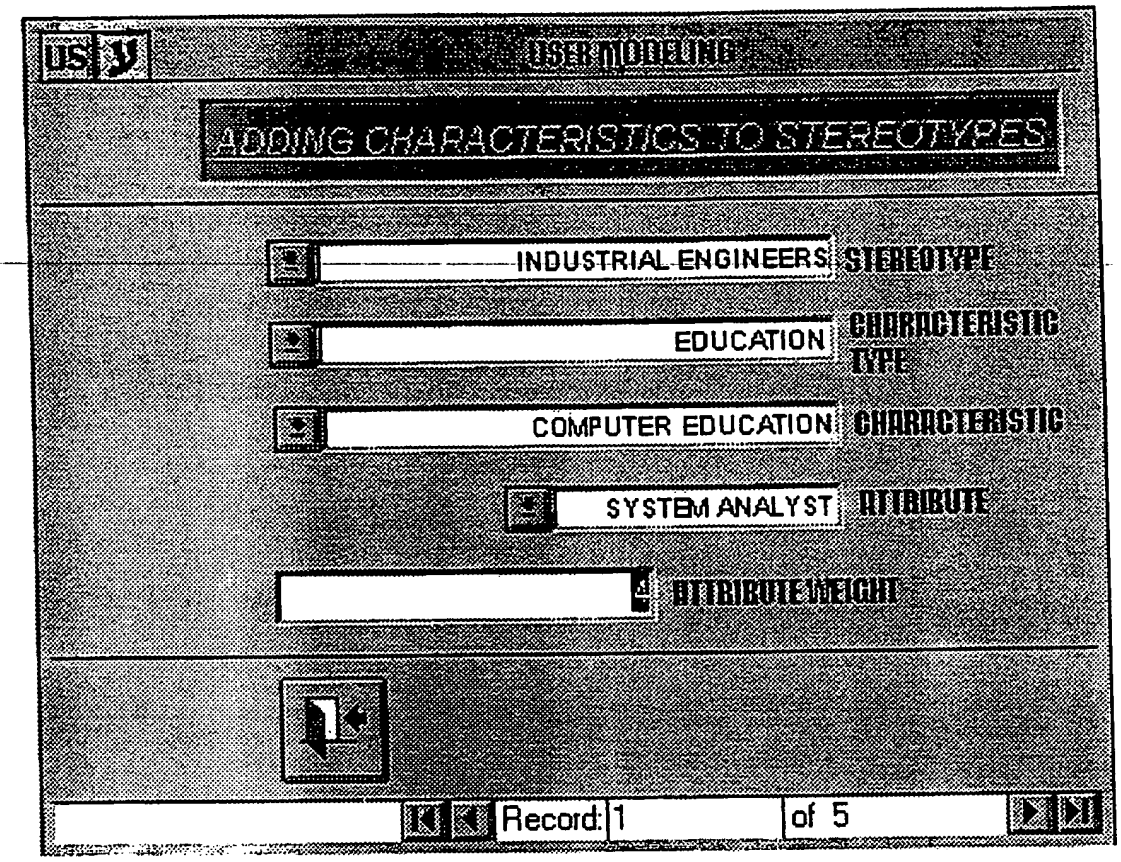

USER MODELING
ADDING CHARACTERISTICS TO STEREOTYPES
INDUSTRIAL ENGINEERS
STEREOTYPE
EDUCATION
CHARACTERISTIC TYPE
COMPUTER EDUCATION
CHARACTERISTIC
SYSTEM ANALYST
ATTRIBUTE
ATTRIBUTE WEIGHT
Record: 1
of 5

Fig. 14E

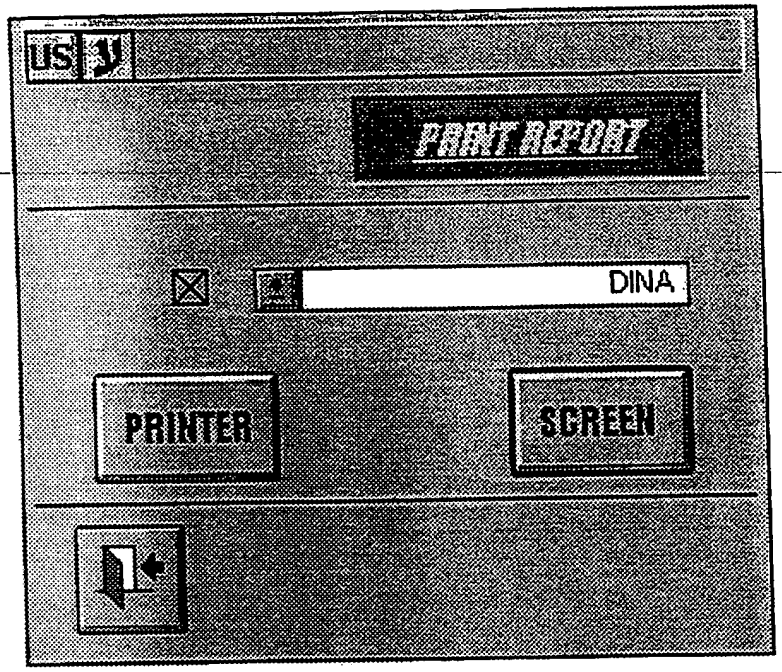


Fig. 14F

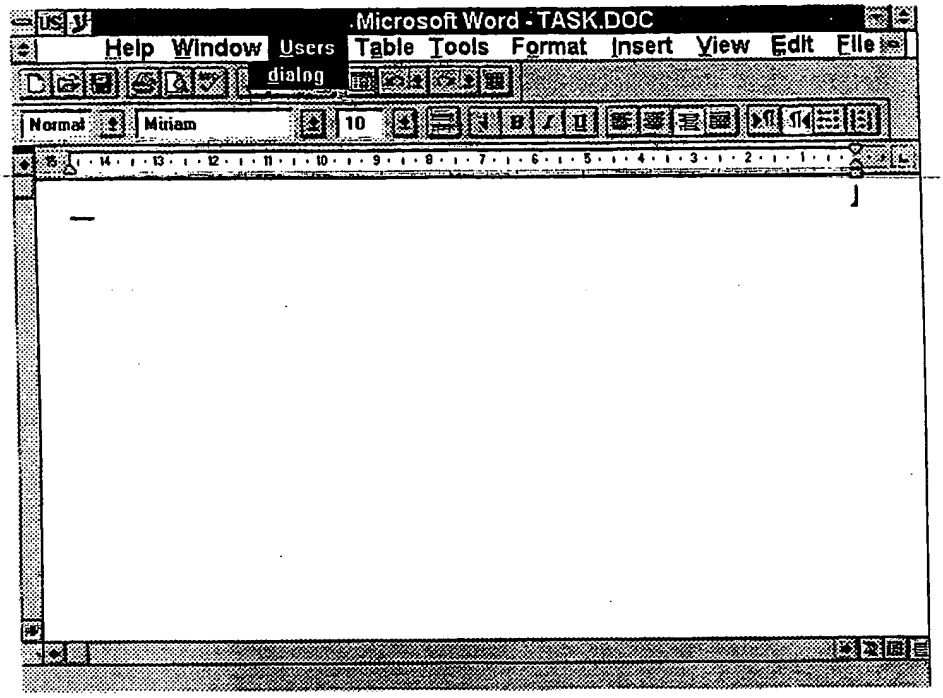

Microsoft Word - TASK.DOC
Help Window Users Table Tools Format Insert View Edit File
dialog
Normal
Miriam
10

Fig. 15A

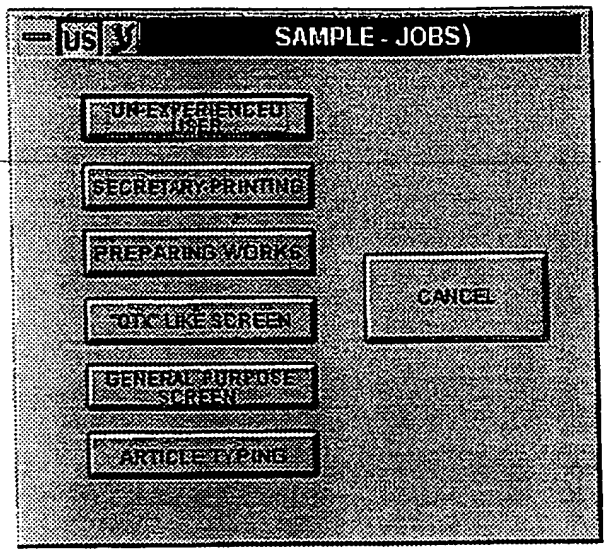

SAMPLE - JOBS)
UNEXPERIENCED USER
SECRETARY-PRINTING
PREPARING WORKS
CANCEL
OFFICE LIKE SCREEN
GENERAL PURPOSE SCREEN
ARTICLE TYPING

Fig. 15B

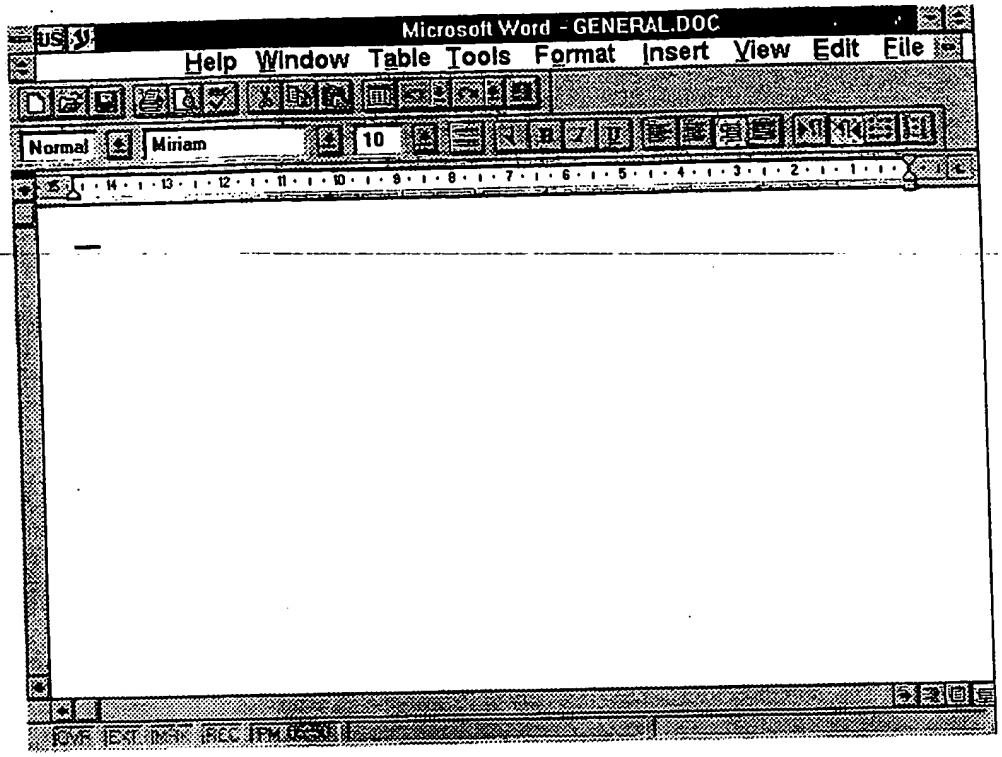

Microsoft Word - GENERAL.DOC
Help Window Table Tools Format Insert View Edit File
Normal
Miriam
10

Fig. 15C

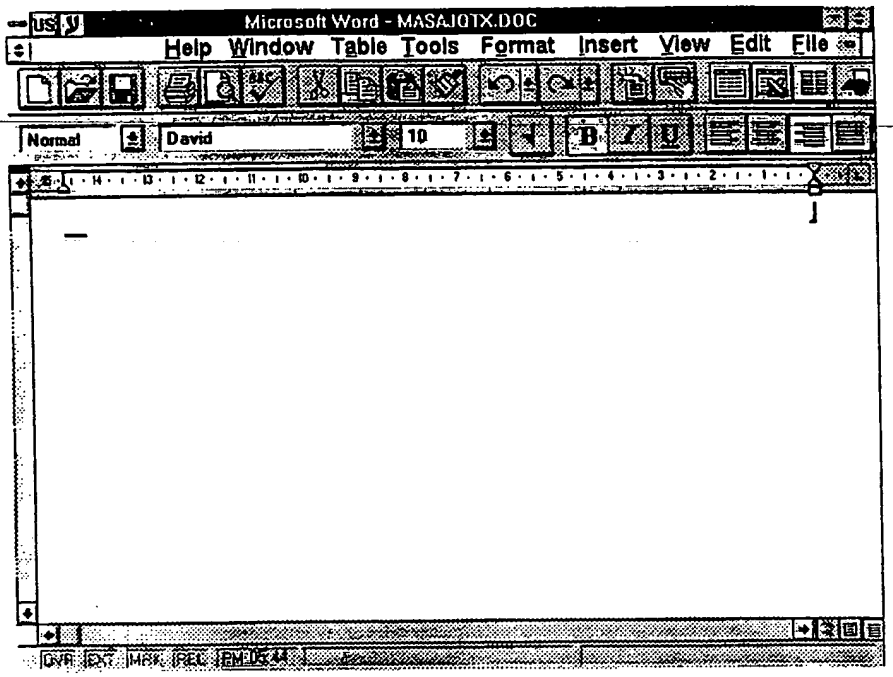

Microsoft Word - MASAJQTX.DOC
Help Window Table Tools Format Insert View Edit File
Normal
David
10

Fig. 15D

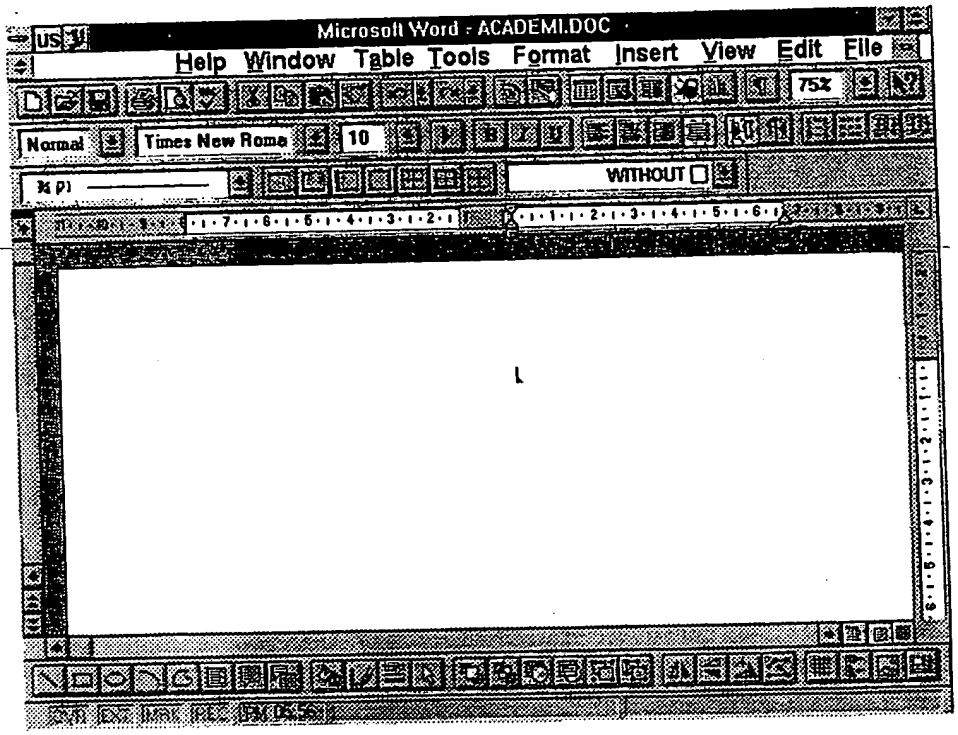

Microsoft Word - ACADEMI.DOC
Help Window Table Tools Format Insert View Edit File
Normal
Times New Roma
10
WITHOUT

Fig. 15E

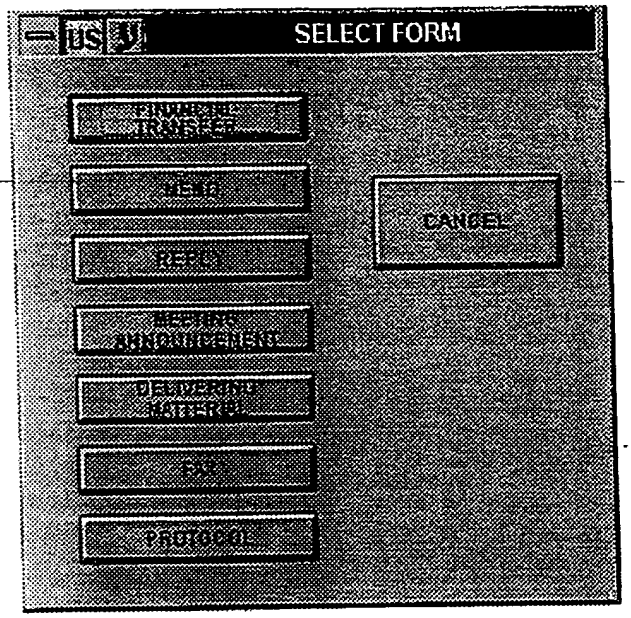


Fig. 15F

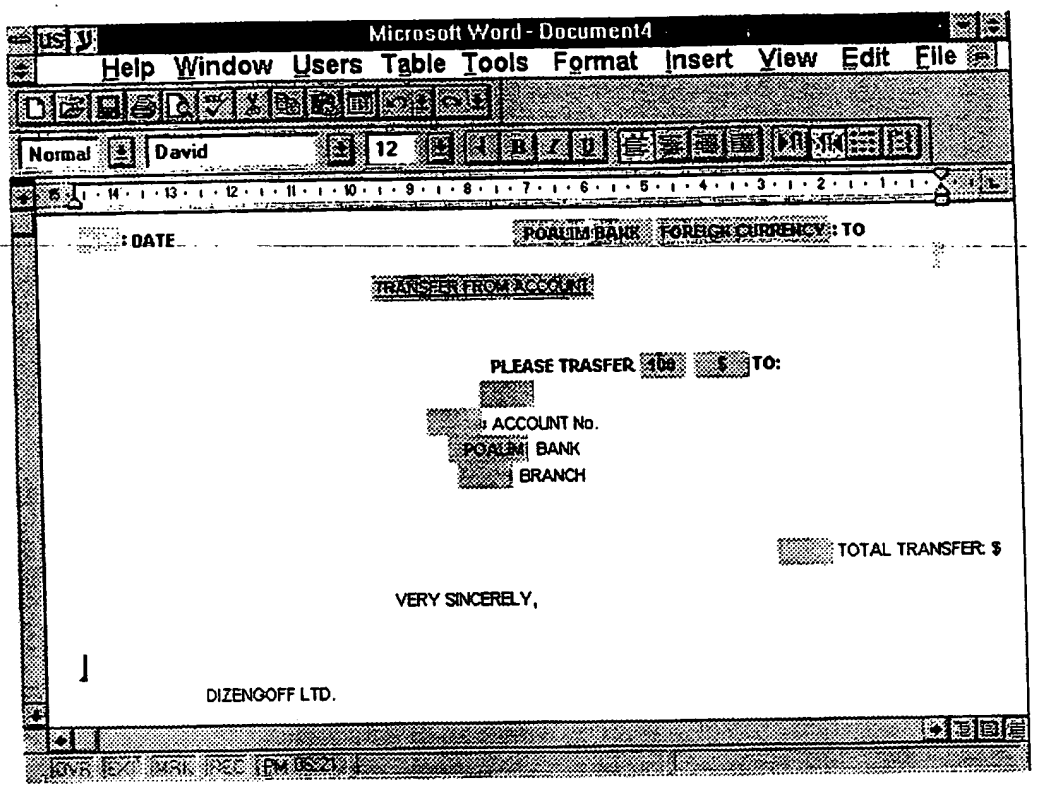

Microsoft Word - Document4
Help Window Users Table Tools Format Insert View Edit File
Normal
David
12
: DATE
POALIM BANK
FOREIGN CURRENCY : TO
TRANSFER FROM ACCOUNT:
PLEASE TRASFER
TO:
ACCOUNT No.
POALIM BANK
BRANCH
TOTAL TRANSFER: $
VERY SINCERELY,
DIZENGOFF LTD.

Fig. 15G

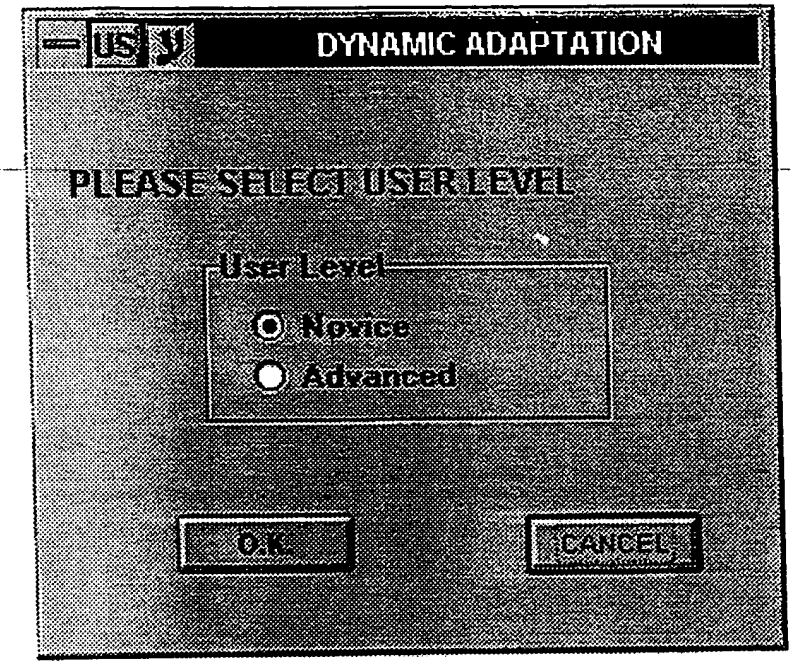

DYNAMIC ADAPTATION
PLEASE SELECT USER LEVEL
User Level
Npvice
Advanced
O.K.
CANCEL

Fig. 16A

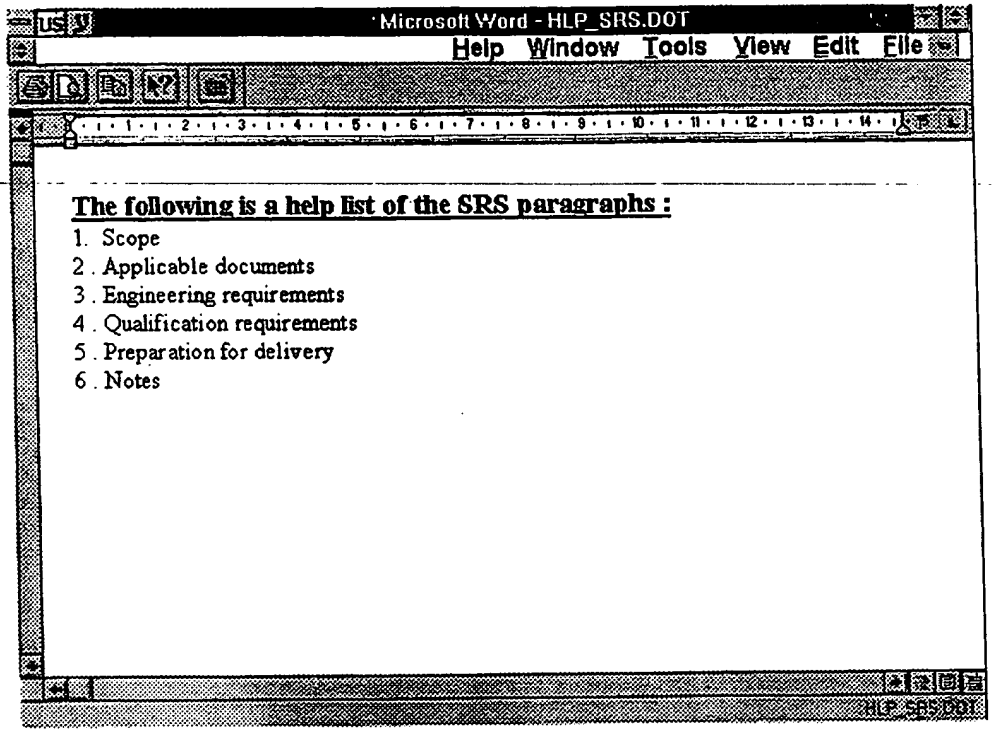

Microsoft Word - HLP_SRS.DOT
Help Window Tools View Edit File
The following is a help list of the SRS paragraphs :
1. Scope
2. Applicable documents
3. Engineering requirements
4. Qualification requirements
5. Preparation for delivery
6. Notes

Fig. 16B

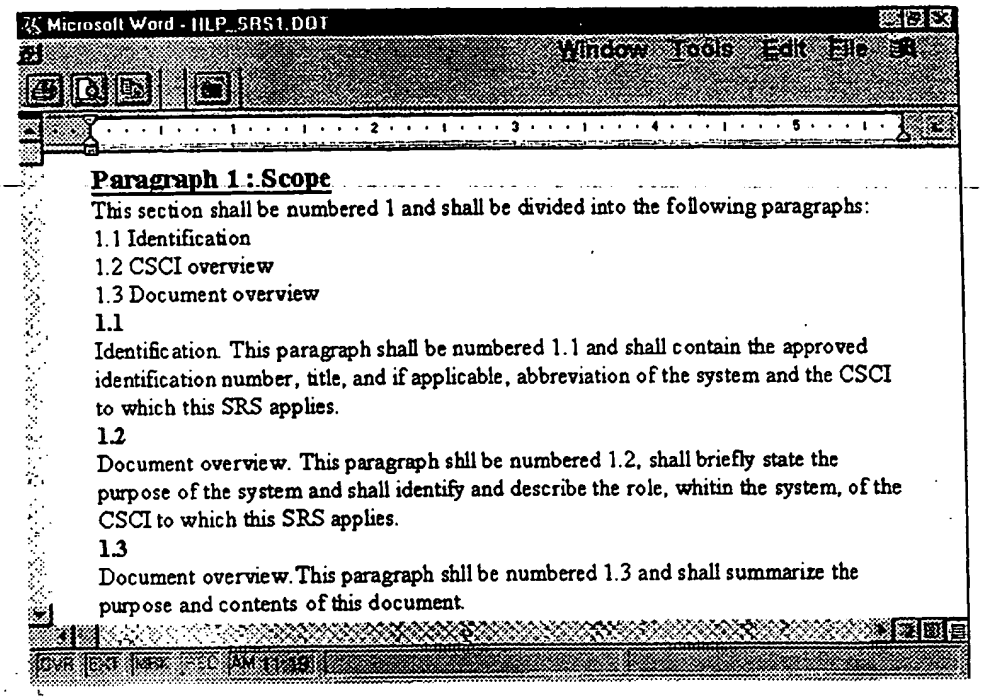

Microsoft Word - IILP_SRS1.DOT
Window Tools Edit File
Paragraph 1 : Scope
This section shall be numbered 1 and shall be divided into the following paragraphs:
1.1 Identification
1.2 CSCI overview
1.3 Document overview
1.1
Identification. This paragraph shall be numbered 1.1 and shall contain the approved identification number, title, and if applicable, abbreviation of the system and the CSCI to which this SRS applies.
1.2
Document overview. This paragraph shll be numbered 1.2, shall briefly state the purpose of the system and shall identify and describe the role, whitin the system, of the CSCI to which this SRS applies.
1.3
Document overview. This paragraph shll be numbered 1.3 and shall summarize the purpose and contents of this document.

Fig. 16C

INTERNATIONAL SEARCH REPORT

International Application No
PC., IL 99/00432

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 98 03907 A (MICROSOFT CORP) 29 January 1998 (1998-01-29) cited in the application the whole document	1-57
Y	WO 92 02880 A (TANDY CORP) 20 February 1992 (1992-02-20) the whole document	1-57
A	US 5 465 358 A (KIEL HARVEY G ET AL) 7 November 1995 (1995-11-07) the whole document	1-57

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
24 November 1999	01/12/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040. Tx. 31 651 epo nl. Fax: (+31-70) 340-3016	Fonderson, A

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
P../IL 99/00432

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	"INTELLIGENT USER INTERFACE PROMPT LEVEL" IBM TECHNICAL DISCLOSURE BULLETIN,US,IBM CORP. NEW YORK, vol. 35, no. 1A, page 25-26 XP000308751 ISSN: 0018-8689 the whole document	1-57

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PC./IL 99/00432

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9803907 A	29-01-1998	CA 2210601 A	19-01-1998
		EP 0912932 A	06-05-1999
WO 9202880 A	20-02-1992	US 5103498 A	07-04-1992
		AU 8401991 A	02-03-1992
		DE 69130766 D	25-02-1999
		DE 69130766 T	02-09-1999
		EP 0541712 A	19-05-1993
		SG 48223 A	17-04-1998
US 5465358 A	07-11-1995	JP 2511642 B	03-07-1996
		JP 7244569 A	19-09-1995

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 9/44

A2

(11) International Publication Number: WO 00/28413

(43) International Publication Date: 18 May 2000 (18.05.00)

(21) International Application Number: PCT/US99/26729

(22) International Filing Date: 10 November 1999 (10.11.99)

(30) Priority Data:
60/115,515 12 November 1998 (12.11.98) US
09/196,479 19 November 1998 (19.11.98) US

(71) Applicant *(for all designated States except US)*: AC PROPERTIES B.V. [NL/NL]; Parkstraat 83, NL-2514 JG 'S Gravenhage (NL).

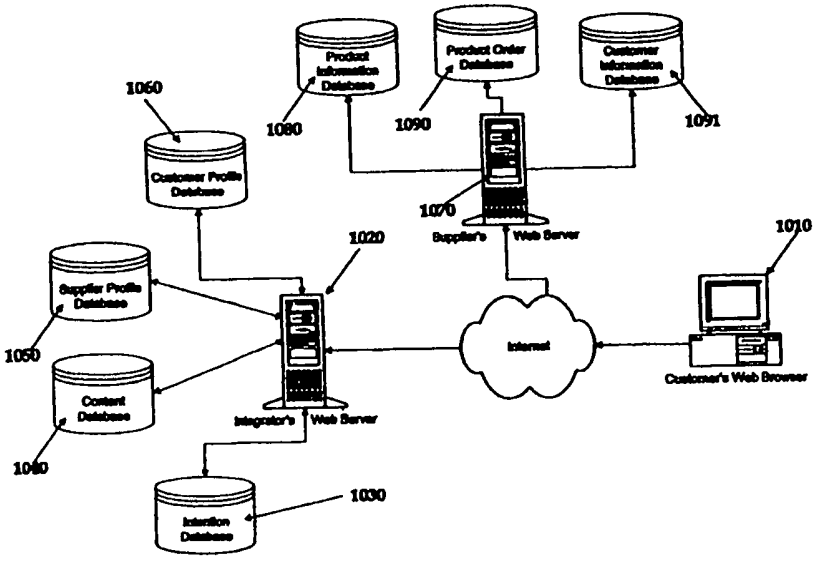
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: HANDEL, Sean, P. [US/US]; 2927 Pine Street, San Francisco, CA 94115 (US). DAY, Brian [US/US]; 1112 Palm Drive, Burlingame, CA 94919 (US). YUEN, Miya [US/US]; 748 Bounty Drive #4802, Foster City, CA 94404 (US).

(74) Agent: STEPHENS, L., Keith; Hickman Stephens & Coleman, LLP, P.O. Box 52037, Palo Alto, CA 94303 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: A SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR A CLIENT INTENTION NETWORKING EXPERIENCE



(57) Abstract

A system is disclosed that facilitates web-based user network interface is created by obtaining user profile information, discerning user intentions regarding a plurality of applications based on the profile information and displaying the plurality of applications formatted in accordance with rules in the user profile information on a display. The user is prompted to provide user profile information that is stored in a database. Then, the system acquires additional information about a user based on the user's interaction with applications and updates the database with the interaction information. The profile information is utilized to provide a common user interface for interacting with one or more applications.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

A SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR A CLIENT INTENTION NETWORKING EXPERIENCE

Field Of The Invention

The present invention relates to agent based systems and more particularly to an agent based system for providing a user interface that facilitates tuning of the user experience to the personal intentions of a particular user.

Agent based technology has become increasingly important for use with applications designed to interact with a user for performing various computer based tasks in foreground and background modes. Agent software comprises computer programs that are set on behalf of users to perform routine, tedious and time-consuming tasks. To be useful to an individual user, an agent must be personalized to the individual user's goals, habits and preferences. Thus, there exists a substantial requirement for the agent to efficiently and effectively acquire user-specific knowledge from the user and utilize it to perform tasks on behalf of the user.

The concept of agency, or the user of agents, is well established. An agent is a person authorized by another person, typically referred to as a principal, to act on behalf of the principal. In this manner the principal empowers the agent to perform any of the tasks that the principal is unwilling or unable to perform. For example, an insurance agent may handle all of the insurance requirements for a principal, or a talent agent may act on behalf of a performer to arrange concert dates.

With the advent of the computer, a new domain for employing agents has arrived. Significant advances in the realm of expert systems enable computer programs to act on behalf of computer users to perform routine, tedious and other time-consuming tasks. These computer programs are referred to as "software agents."

Moreover, there has been a recent proliferation of computer and communication networks. These networks permit a user to access vast amounts of information and services without, essentially, any geographical boundaries. Thus, a software agent has a rich environment to perform a large number of tasks on behalf of a user. For example, it is now possible for an agent to make an airline reservation, purchase the ticket, and have the ticket delivered directly to a user. Similarly, an agent could scan the Internet and obtain information ranging from the latest sports or news to a particular graduate thesis in applied physics. Current solutions fail to apply agent technology to existing calendar technology to provide targeted acquisition of background information for a user's upcoming events.

SUMMARY OF THE INVENTION

According to a broad aspect of a preferred embodiment of the invention, a user network interface is created by obtaining user profile information, discerning user intentions regarding a plurality of applications based on the profile information and displaying the plurality of applications formatted in accordance with rules in the user profile information on a display. The user is prompted to provide user profile information that is stored in a database. Then, the system acquires additional information about a user based on the user's interaction with applications and updates the database with the interaction information. The profile information is utilized to provide a common user interface for interacting with one or more applications.

DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, aspects and advantages are better understood from the following detailed description of a preferred embodiment of the invention with reference to the drawings, in which:

Figure 1 is a block diagram of a representative hardware environment in accordance with a preferred embodiment;

Figure 2 is a flowchart of the system in accordance with a preferred embodiment;

Figure 3 is a flowchart of a parsing unit of the system in accordance with a preferred embodiment;

Figure 4 is a flowchart for pattern matching in accordance with a preferred embodiment;

Figures 5 is a flowchart for a search unit in accordance with a preferred embodiment;

Figure 6 is a flowchart for overall system processing in accordance with a preferred embodiment;

Figure 7 is a flowchart of topic processing in accordance with a preferred embodiment;

Figure 8 is a flowchart of meeting record processing in accordance with a preferred embodiment;

Figure 9 is a block diagram of process flow of a pocket bargain finder in accordance with a preferred embodiment;

Figure 10A and 10B are a block diagram and flowchart depicting the logic associated with creating a customized content web page in accordance with a preferred embodiment;

Figure 11 is a flowchart depicting the detailed logic associated with retrieving user-centric content in accordance with a preferred embodiment;

Figure 12 is a data model of a user profile in accordance with a preferred embodiment;

Figure 13 is a persona data model in accordance with a preferred embodiment;

Figure 14 is an intention data model in accordance with a preferred embodiment;

Figure 15 is a flowchart of the processing for generating an agent's current statistics in accordance with a preferred embodiment;

Figure 16 is a flowchart of the logic that determines the personalized product rating for a user in accordance with a preferred embodiment;

Figure 17 is a flowchart of the logic for accessing the centrally stored profile in accordance with a preferred embodiment;

Figure 18 is a flowchart of the interaction logic between a user and the integrator for a particular supplier in accordance with a preferred embodiment;

Figure 19 is a flowchart of the agent processing for generating a verbal summary in accordance with a preferred embodiment;

Figure 20 illustrates a display login in accordance with a preferred embodiment;

Figure 21 illustrates a managing daily logistics display in accordance with a preferred embodiment;

Figure 22 illustrates a user main display in accordance with a preferred embodiment;

Figure 23 illustrates an agent interaction display in accordance with a preferred embodiment;

Figure 24 is a block diagram of an active knowledge management system in accordance with a preferred embodiment;

Figure 25 is a block diagram of a back end server in accordance with a preferred embodiment; and

Figure 26 is a block diagram of a magic wall in accordance with a preferred embodiment.

DETAILED DESCRIPTION

A preferred embodiment of a system in accordance with the present invention is preferably practiced in the context of a personal computer such as an IBM compatible personal computer, Apple Macintosh computer or UNIX based workstation. A representative hardware environment is depicted in Figure 1, which illustrates a typical hardware configuration of a workstation in accordance with a preferred embodiment having a central processing unit 110, such as a microprocessor, and a number of other units interconnected via a system bus 112. The workstation shown in Figure 1 includes a Random Access Memory (RAM) 114, Read Only Memory (ROM) 116, an I/O adapter 118 for connecting peripheral devices such as disk storage units 120 to the bus 112, a user interface adapter 122 for connecting a keyboard 124, a mouse 126, a speaker 128, a microphone 132, and/or other user interface devices such as a touch screen (not shown) to the bus 112, communication adapter 134 for connecting the workstation to a communication network (e.g., a data processing network) and a display adapter 136 for connecting the bus 112 to a display device 138. The workstation typically has resident thereon an operating system such as the Microsoft Windows NT or Windows/95 Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. Those skilled in the art will appreciate that the present invention may also be implemented on platforms and operating systems other than those mentioned.

A preferred embodiment is written using JAVA, C, and the C++ language and utilizes object oriented programming methodology. Object oriented programming (OOP) has become increasingly used to develop complex applications. As OOP moves toward the mainstream of software design and development, various software solutions require adaptation to make use of the benefits of OOP. A need exists for these principles of OOP to be applied to a messaging interface of an electronic messaging system such that a set of OOP classes and objects for the messaging interface can be provided.

OOP is a process of developing computer software using objects, including the steps of analyzing the problem, designing the system, and constructing the program. An object is a software package that contains both data and a collection of related structures and procedures. Since it contains both data and a collection of structures and procedures, it can be visualized as a self-sufficient component that does not require other additional structures, procedures or data to perform its specific task. OOP, therefore, views a computer program as a collection of largely autonomous components, called objects, each of which is responsible for a specific task. This concept of packaging data, structures, and procedures together in one component or module is called encapsulation.

In general, OOP components are reusable software modules which present an interface that conforms to an object model and which are accessed at run-time through a component integration architecture. A component integration architecture is a set of architecture mechanisms which allow software modules in different process spaces to utilize each others capabilities or functions. This is generally done by assuming a common component object model on which to build the architecture.

It is worthwhile to differentiate between an object and a class of objects at this point. An object is a single instance of the class of objects, which is often just called a class. A class of objects can be viewed as a blueprint, from which many objects can be formed.

OOP allows the programmer to create an object that is a part of another object. For example, the object representing a piston engine is said to have a composition-relationship with the object representing a piston. In reality, a piston engine comprises a piston, valves and many other components; the fact that a piston is an element of a piston engine can be logically and semantically represented in OOP by two objects.

OOP also allows creation of an object that "depends from" another object. If there are two objects, one representing a piston engine and the other representing a piston engine wherein the piston is made of ceramic, then the relationship between the two objects is not that of composition. A ceramic piston engine does not make up a piston engine. Rather it is merely one kind of piston engine that has one more limitation than the piston engine; its piston is made of ceramic. In this case, the object representing the ceramic piston engine is called a derived object, and it inherits all of the aspects of the object representing the piston engine and adds further limitation or detail to it. The object representing the ceramic piston engine "depends from" the object representing the piston engine. The relationship between these objects is called inheritance.

When the object or class representing the ceramic piston engine inherits all of the aspects of the objects representing the piston engine, it inherits the thermal characteristics of a standard piston defined in the piston engine class. However, the ceramic piston engine object overrides these ceramic specific thermal characteristics, which are typically different from those associated

with a metal piston. It skips over the original and uses new functions related to ceramic pistons. Different kinds of piston engines have different characteristics, but may have the same underlying functions associated with it (e.g., how many pistons in the engine, ignition sequences, lubrication, etc.). To access each of these functions in any piston engine object, a programmer would call the same functions with the same names, but each type of piston engine may have different/overriding implementations of functions behind the same name. This ability to hide different implementations of a function behind the same name is called polymorphism and it greatly simplifies communication among objects.

With the concepts of composition-relationship, encapsulation, inheritance and polymorphism, an object can represent just about anything in the real world. In fact, our logical perception of the reality is the only limit on determining the kinds of things that can become objects in object-oriented software. Some typical categories are as follows:

- Objects can represent physical objects, such as automobiles in a traffic-flow simulation, electrical components in a circuit-design program, countries in an economics model, or aircraft in an air-traffic-control system.
- Objects can represent elements of the computer-user environment such as windows, menus or graphics objects.
- An object can represent an inventory, such as a personnel file or a table of the latitudes and longitudes of cities.
- An object can represent user-defined data types such as time, angles, and complex numbers, or points on the plane.

With this enormous capability of an object to represent just about any logically separable matters, OOP allows the software developer to design and implement a computer program that is a model of some aspects of reality, whether that reality is a physical entity, a process, a system, or a composition of matter. Since the object can represent anything, the software developer can create an object which can be used as a component in a larger software project in the future.

If 90% of a new OOP software program consists of proven, existing components made from preexisting reusable objects, then only the remaining 10% of the new software project has to be written and tested from scratch. Since 90% already came from an inventory of extensively tested reusable objects, the potential domain from which an error could originate is 10% of the program. As a result, OOP enables software developers to build objects out of other, previously built, objects.

This process closely resembles complex machinery being built out of assemblies and sub-assemblies. OOP technology, therefore, makes software engineering more like hardware engineering in that software is built from existing components, which are available to the developer as objects. All this adds up to an improved quality of the software as well as an increased speed of its development.

Programming languages are beginning to fully support the OOP principles, such as encapsulation, inheritance, polymorphism, and composition-relationship. With the advent of the C++ language, many commercial software developers have embraced OOP. C++ is an OOP language that offers a fast, machine-executable code. Furthermore, C++ is suitable for both commercial-application and systems-programming projects. For now, C++ appears to be the most popular choice among many OOP programmers, but there is a host of other OOP languages, such as Smalltalk, common lisp object system (CLOS), and Eiffel. Additionally, OOP capabilities are being added to more traditional popular computer programming languages such as Pascal.

The benefits of object classes can be summarized, as follows:

- *Objects* and their corresponding classes break down complex programming problems into many smaller, simpler problems.
- *Encapsulation* enforces data abstraction through the organization of data into small, independent objects that can communicate with each other. Encapsulation protects the data in an object from accidental damage, but allows other objects to interact with that data by calling the object's member functions and structures.
- *Subclassing* and inheritance make it possible to extend and modify objects through deriving new kinds of objects from the standard classes available in the system. Thus, new capabilities are created without having to start from scratch.
- *Polymorphism* and multiple inheritance make it possible for different programmers to mix and match characteristics of many different classes and create specialized objects that can still work with related objects in predictable ways.
- *Class hierarchies* and containment hierarchies provide a flexible mechanism for modeling real-world objects and the relationships among them.
- *Libraries* of reusable classes are useful in many situations, but they also have some limitations. For example:
- *Complexity.* In a complex system. the class hierarchies for related classes can become extremely confusing, with many dozens or even hundreds of classes.
- *Flow of control.* A program written with the aid of class libraries is still responsible for the flow of control (i.e., it must control the interactions among all the objects created from a particular library). The programmer has to decide which functions to call at what times for which kinds of objects.
- *Duplication of effort.* Although class libraries allow programmers to use and reuse many small pieces of code, each programmer puts those pieces together in a different way. Two different programmers can use the same set of class libraries to write two programs that do exactly the same thing but whose internal structure (i.e., design) may be quite different, depending on hundreds of small decisions each programmer makes along the way. Inevitably, similar pieces of code end up doing similar things in slightly different ways and do not work as well together as they should.

Class libraries are very flexible. As programs grow more complex, more programmers are forced to reinvent basic solutions to basic problems over and over again. A relatively new extension of the class library concept is to have a framework of class libraries. This framework is more complex and consists of significant collections of collaborating classes that capture both the small scale patterns and major mechanisms that implement the common requirements and design in a specific application domain. They were first developed to free application programmers from the chores involved in displaying menus, windows, dialog boxes, and other standard user interface elements for personal computers.

Frameworks also represent a change in the way programmers think about the interaction between the code they write and code written by others. In the early days of procedural programming, the programmer called libraries provided by the operating system to perform certain tasks, but basically the program executed down the page from start to finish, and the programmer was solely responsible for the flow of control. This was appropriate for printing out paychecks, calculating a mathematical table, or solving other problems with a program that executed in just one way.

The development of graphical user interfaces began to turn this procedural programming arrangement inside out. These interfaces allow the user, rather than program logic, to drive the program and decide when certain actions should be performed. Today, most personal computer software accomplishes this by means of an event loop which monitors the mouse, keyboard, and other sources of external events and calls the appropriate parts of the programmer's code according to actions that the user performs. The programmer no longer determines the order in which events occur. Instead, a program is divided into separate pieces that are called at unpredictable times and in an unpredictable order. By relinquishing control in this way to users, the developer creates a program that is much easier to use. Nevertheless, individual pieces of the program written by the developer still call libraries provided by the operating system to accomplish certain tasks, and the programmer must still determine the flow of control within each piece after being called by the event loop. Application code still "sits on top of" the system.

Even event loop programs require programmers to write a lot of code that should not need to be written separately for every application. The concept of an application framework carries the event loop concept further. Instead of dealing with all the nuts and bolts of constructing basic menus, windows, and dialog boxes and then making these things all work together, programmers using application frameworks start with working application code and basic user interface elements in place. Subsequently, they build from there by replacing some of the generic capabilities of the framework with the specific capabilities of the intended application.

Application frameworks reduce the total amount of code that a programmer has to write from scratch. However, because the framework is really a generic application that displays windows, supports copy and paste, and so on, the programmer can also relinquish control to a greater degree than event loop programs permit. The framework code takes care of almost all event handling and flow of control, and the programmer's code is called only when the framework needs it (e.g., to create or manipulate a proprietary data structure).

A programmer writing a framework program not only relinquishes control to the user (as is also true for event loop programs), but also relinquishes the detailed flow of control within the program to the framework. This approach allows the creation of more complex systems that work together in interesting ways, as opposed to isolated programs, having custom code, being created over and over again for similar problems.

Thus, as is explained above, a framework basically is a collection of cooperating classes that make up a reusable design solution for a given problem domain. It typically includes objects that provide default behavior (e.g., for menus and windows), and programmers use it by inheriting some of that default behavior and overriding other behavior so that the framework calls application code at the appropriate times.

There are three main differences between frameworks and class libraries:

- *Behavior versus protocol.* Class libraries are essentially collections of behaviors that you can call when you want those individual behaviors in your program. A framework, on the other hand, provides not only behavior but also the protocol or set of rules that govern the ways in which behaviors can be combined, including rules for what a programmer is supposed to provide versus what the framework provides.

- *Call versus override.* With a class library, the code the programmer instantiates objects and calls their member functions. It's possible to instantiate and call objects in the same way with a framework (i.e., to treat the framework as a class library), but to take full advantage of a framework's reusable design, a programmer typically writes code that overrides and is called by the framework. The framework manages the flow of control among its objects. Writing a program involves dividing responsibilities among the various pieces of software that are called by the framework rather than specifying how the different pieces should work together.
- *Implementation versus design.* With class libraries, programmers reuse only implementations, whereas with frameworks, they reuse design. A framework embodies the way a family of related programs or pieces of software work. It represents a generic design solution that can be adapted to a variety of specific problems in a given domain. For example, a single framework can embody the way a user interface works, even though two different user interfaces created with the same framework might solve quite different interface problems.

Thus, through the development of frameworks for solutions to various problems and programming tasks, significant reductions in the design and development effort for software can be achieved. A preferred embodiment of the invention utilizes HyperText Markup Language (HTML) to implement documents on the Internet together with a general-purpose secure communication protocol for a transport medium between the client and the Newco. HTTP or other protocols could be readily substituted for HTML without undue experimentation. Information on these products is available in T. Berners-Lee, D. Connoly, "RFC 1866: Hypertext Markup Language - 2.0" (Nov. 1995); and R. Fielding, H, Frystyk, T. Berners-Lee, J. Gettys and J.C. Mogul, "Hypertext Transfer Protocol – HTTP/1.1: HTTP Working Group Internet Draft" (May 2, 1996). HTML is a simple data format used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of domains. HTML has been in use by the World-Wide Web global information initiative since 1990. HTML is an application of ISO Standard 8879:1986 Information Processing Text and Office Systems; Standard Generalized Markup Language (SGML).

To date, Web development tools have been limited in their ability to create dynamic Web applications which span from client to server and interoperate with existing computing resources. Until recently, HTML has been the dominant technology used in development of Web-based solutions. However, HTML has proven to be inadequate in the following areas:

- Poor performance;
- Restricted user interface capabilities;
- Can only produce static Web pages;
- Lack of interoperability with existing applications and data; and
- Inability to scale.

Sun Microsystem's Java language solves many of the client-side problems by:

- Improving performance on the client side;
- Enabling the creation of dynamic, real-time Web applications; and
- Providing the ability to create a wide variety of user interface components.

With Java, developers can create robust User Interface (UI) components. Custom "widgets" (e.g. real-time stock tickers, animated icons, etc.) can be created, and client-side performance is improved. Unlike HTML, Java supports the notion of client-side validation, offloading appropriate processing onto the client for improved performance. Dynamic, real-time Web pages can be created. Using the above-mentioned custom UI components, dynamic Web pages can also be created.

Sun's Java language has emerged as an industry-recognized language for "programming the Internet." Sun defines Java as: "a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, dynamic, buzzword-compliant, general-purpose programming language. Java supports programming for the Internet in the form of platform-independent Java applets." Java applets are small, specialized applications that comply with Sun's Java Application Programming Interface (API) allowing developers to add "interactive content" to Web documents (e.g. simple animations, page adornments, basic games, etc.). Applets execute within a Java-compatible browser (e.g. Netscape Navigator) by copying code from the server to client. From a language standpoint, Java's core feature set is based on C++. Sun's Java literature states that Java is basically "C++, with extensions from Objective C for more dynamic method resolution".

Another technology that provides similar function to JAVA is provided by Microsoft and ActiveX Technologies, to give developers and Web designers wherewithal to build dynamic content for the Internet and personal computers. ActiveX includes tools for developing animation, 3-D virtual reality, video and other multimedia content. The tools use Internet standards, work on multiple platforms, and are being supported by over 100 companies. The group's building blocks are called ActiveX Controls, small, fast components that enable developers to embed parts of software in hypertext markup language (HTML) pages. ActiveX Controls work with a variety of programming languages including Microsoft Visual C++, Borland Delphi, Microsoft Visual Basic programming system and, in the future, Microsoft's development tool for Java, code named "Jakarta." ActiveX Technologies also includes ActiveX Server Framework, allowing developers to create server applications. One of ordinary skill in the art readily recognizes that ActiveX could be substituted for JAVA without undue experimentation to practice the invention.

In accordance with a preferred embodiment, BackgroundFinder (BF) is implemented as an agent responsible for preparing an individual for an upcoming meeting by helping him/her retrieve relevant information about the meeting from various sources. BF receives input text in character form indicative of the target meeting. The input text is generated in accordance with a preferred embodiment by a calendar program that includes the time of the meeting. As the time of the meeting approaches, the calendar program is queried to obtain the text of the target event and that information is utilized as input to the agent. Then, the agent parses the input meeting text to extract its various components such as title, body, participants, location, time etc. The system also performs pattern matching to identify particular meeting fields in a meeting text. This information is utilized to query various sources of information on the web and obtain relevant stories about the current meeting to send back to the calendaring system. For example, if an individual has a meeting with Netscape and Microsoft to talk about their disputes, and would obtain this initial information from the calendaring system. It will then parse out the text to realize that the companies in the meeting are "Netscape" and "Microsoft" and the topic is "disputes." Then, the system queries the web for relevant information concerning the topic. Thus, in accordance with an objective of the invention, the system updates the calendaring system and eventually the user with the best information it can gather to prepare the user for the target meeting. In accordance with a preferred embodiment, the information is stored in a file that is obtained via selection from a link imbedded in the calendar system.

PROGRAM ORGANIZATION

A computer program in accordance with a preferred embodiment is organized in five distinct modules: BF.Main, BF.Parse, Background Finder.Error, BF.PatternMatching and BF.Search. There is also a frmMain which provides a user interface used only for debugging purposes. The executable programs in accordance with a preferred embodiment never execute with the user interface and should only return to the calendaring system through Microsoft's Winsock control. A preferred embodiment of the system executes in two different modes which can be specified under the command line sent to it by the calendaring system. When the system runs in simple mode, it executes a keyword query to submit to external search engines. When executed in complex mode, the system performs pattern matching before it forms a query to be sent to a search engine.

DATA STRUCTURES

The system in accordance with a preferred embodiment utilizes three user defined structures:

1. TMeetingRecord;
2. TPatternElement; and
3. TPatternRecord.

The user-defined structure, tMeetingRecord, is used to store all the pertinent information concerning a single meeting. This info includes userID, an original description of the meeting, the extracted list of keywords from the title and body of meeting etc. It is important to note that only one meeting record is created per instance of the system in accordance with a preferred embodiment. This is because each time the system is spawned to service an upcoming meeting, it is assigned a task to retrieve information for only one meeting. Therefore, the meeting record created corresponds to the current meeting examined. ParseMeetingText populates this meeting record and it is then passed around to provide information about the meeting to other functions. If GoPatternMatch can bind any values to a particular meeting field, the corresponding entries in the meeting record is also updated. The structure of tMeetingRecord with each field described in parentheses is provided below in accordance with a preferred embodiment.

```
A.1.1.1.1.1                 Public Type tMeetingRecord
  sUserID As String         (user id given by Munin)
  sTitleOrig As String      (original non stop listed title we need to keep around to send back to Munin)
  sTitleKW As String        (stoplisted title with only keywords)
  sBodyKW As String         (stoplisted body with only keywords)
  sCompany() As String      (companys identified in title or body through pattern matching)
  sTopic() As String        (topics identified in title or body through pattern matching)
  sPeople() As String       (people identified in title or body through pattern matching)
  sWhen() As String         (time identified in title or body through pattern matching)
  sWhere() As String        (location identified in title or body through pattern matching)
  sLocation As String       (location as passed in by Munin)
  sTime As String           (time as passed in by Munin)
  sParticipants() As String (all participants engaged as passed in by Munin)
  sMeetingText As String    (the original meeting text w/o userid)
End Type
```

There are two other structures which are created to hold each individual pattern utilized in pattern matching. The record tAPatternRecord is an array containing all the components / elements of a pattern. The type tAPatternElement is an array of strings which represent an element in a pattern. Because there may be many "substitutes" for each element, we need an array of strings to keep track of what all the substitutes are. The structures of tAPatternElement and tAPatternRecord are presented below in accordance with a preferred embodiment.

```
Public Type tAPatternElement
  elementArray() As String
End Type
Public Type tAPatternRecord
  patternArray() As tAPatternElement
End Type
```

COMMON USER DEFINED CONSTANTS

Many constants are defined in each declaration section of the program which may need to be updated periodically as part of the process of maintaining the system in accordance with a preferred embodiment. The constants are accessible to allow dynamic configuration of the system to occur as updates for maintaining the code.

Included in the following tables are lists of constants from each module which I thought are most likely to be modified from time to time. However, there are also other constants used in the code not included in the following list. It does not mean that these non-included constants will never be changed. It means that they will change much less frequently.

For the Main Module (BF.Main) :

CONSTANT	PRESET VALUE	USE
MSGTOMUNIN_TYPE	6	Define the message number used to identify messages between BF and Munin
IP_ADDRESS_MUNIN	"10.2.100.48"	Define the IP address of the machine in which Munin and BF are running on so they can transfer data through UDP.
PORT_MUNIN	7777	Define the remote port in which we are operating on.
TIMEOUT_AV	60	Define constants for setting time out in inet controls
TIMEOUT_NP	60	Define constants for setting time out in inet controls
CMD_SEPARATOR	"\"	Define delimiter to tell which part of Munin's command represents the beginning of our input meeting text

CONSTANT	PRESET VALUE	USE
OUTPARAM_SEPARATOR	"::"	Define delimiter for separating out different portions of the output. The separator is for delimiting the msg type, the user id, the meeting title and the beginning of the actual stories retrieved.

For the Search Module (BF.Search):

CONSTANT	CURRENT VALUE	USE
PAST_NDAYS	5	Define number of days you want to look back for AltaVista articles. Doesn't really matter now because we aren't really doing a news search in alta vista. We want all info.
CONNECTOR_AV_URL	"+AND+"	Define how to connect keywords. We want all our keywords in the string so for now use AND. If you want to do an OR or something, just change connector.
CONNECTOR_NP_URL	"+AND+"	Define how to connect keywords. We want all our keywords in the string so for now use AND. If you want to do an OR or something, just change connector.
NUM_NP_STORIES	3	Define the number of stories to return back to Munin from NewsPage.
NUM_AV_STORIES	3	Define the number of stories to return back to Munin from AltaVista.

5

For the Parse Module (BF.Parse):

CONSTANT	CURRENT VALUE	USE
PORTION_SEPARATOR	"::"	Define the separator between different portions of the meeting text sent in by Munin. For example in "09::Meet with Chad::about life::Chad | Denise::::::" "::" is the separator

CONSTANT	CURRENT VALUE	USE
		between different parts of the meeting text.
PARTICIPANT_SEPARATOR	"|"	Define the separator between each participant in the participant list portion of the original meeting text. Refer to example above.

For Pattern Matching Module (BFPatternMatch): There are no constants in this module which require frequent updates.

General Process Flow

The best way to depict the process flow and the coordination of functions between each other is with the five flowcharts illustrated in Figures 2 to 6. Figure 2 depicts the overall process flow in accordance with a preferred embodiment. Processing commences at the top of the chart at function block 200 which launches when the program starts. Once the application is started, the command line is parsed to remove the appropriate meeting text to initiate the target of the background find operation in accordance with a preferred embodiment as shown in function block 210. A global stop list is generated after the target is determined as shown in function block 220. Then, all the patterns that are utilized for matching operations are generated as illustrated in function block 230. Then, by tracing through the chart, function block 200 invokes GoBF 240 which is responsible for logical processing associated with wrapping the correct search query information for the particular target search engine. For example, function block 240 flows to function block 250 and it then calls GoPatternMatch as shown in function block 260. To see the process flow of GoPatternMatch, we swap to the diagram titled "Process Flow for BF's Pattern Matching Unit."

One key thing to notice is that functions depicted at the same level of the chart are called by in sequential order from left to right (or top to bottom) by their common parent function. For example, Main 200 calls ProcessCommandLine 210, then CreateStopListist 220, then CreatePatterns 230, then GoBackgroundFinder 240. Figures 3 to 6 detail the logic for the entire program, the parsing unit, the pattern matching unit and the search unit respectively. Figure 6 details the logic determinative of data flow of key information through BackgroundFinder, and shows the functions that are responsible for creating or processing such information.

DETAILED SEARCH ARCHITECTURE UNDER THE SIMPLE QUERY MODE

SEARCH ALTA VISTA

(Function block 270 of Figure 2)

The Alta Vista search engine utilizes the identifies and returns general information about topics related to the current meeting as shown in function block 270 of Figure 2. The system in accordance with a preferred embodiment takes all the keywords from the title portion of the original meeting text and constructs an advanced query to send to Alta Vista. The keywords are logically combined together in the query. The results are also ranked based on the same set of keywords. One of ordinary skill in the art will readily comprehend that a date restriction or publisher criteria could be facilitated on the articles we want to retrieve. A set of top ranking stories are returned to the calendaring system in accordance with a preferred embodiment.

NEWS PAGE

(Function block 275 of Figure 2)

The NewsPage search system is responsible for giving us the latest news topics related to a target meeting. The system takes all of the keywords from the title portion of the original meeting text and constructs a query to send to the NewsPage search engine. The keywords are logically combined together in the query. Only articles published recently are retrieved. The Newspage search system provides a date restriction criteria that is settable by a user according to the user's preference. The top ranking stories are returned to the calendaring system.

Figure 3 is a user profile data model in accordance with a preferred embodiment. Processing commences at function block 300 which is responsible for invoking the program from the main module. Then, at function block 310, a wrapper function is invoked to prepare for the keyword extraction processing in function block 320. After the keywords are extracted, then processing flows to function block 330 to determine if the delimiters are properly positioned. Then, at function block 340, the number of words in a particular string is calculated and the delimiters for the particular field are and a particular field from the meeting text is retrieved at function block 350. Then, at function block 380, the delimiters of the string are again checked to assure they are placed appropriately. Finally, at function block 360, the extraction of each word from the title and body of the message is performed a word at a time utilizing the logic in function block 362 which finds the next closest word delimiter in the input phrase, function block 364 which strips unnecessary materials from a word and function block 366 which determines if a word is on the stop list and returns an error if the word is on the stop list.

PATTERN MATCHING IN ACCORDANCE WITH A PREFERRED EMBODIMENT

The limitations associated with a simple searching method include the following:

1. Because it relies on a stoplist of unwanted words in order to extract from the meeting text a set of keywords, it is limited by how comprehensive the stoplist is. Instead of trying to figure out what parts of the meeting text we should throw away, we should focus on what parts of the meeting text we want.
2. A simple search method in accordance with a preferred embodiment only uses the keywords from a meeting title to form queries to send to Alta Vista and NewsPage. This ignores an alternative source of information for the query, the body of the meeting notice. We cannot include the keywords from the meeting body to form our queries because this often results in queries which are too long and so complex that we often obtain no meaningful results.
3. There is no way for us to tell what each keyword represents. For example, we may extract "Andy" and "Grove" as two keywords. However, a simplistic search has no way knowing that "Andy Grove" is in fact a person's name. Imagine the possibilities if we could somehow intelligently guess that "Andy Grove" is a person's name. We can find out if he is an Andersen person and if so what kind of projects he's been on before etc. etc.
4. In summary, by relying solely on a stoplist to parse out unnecessary words, we suffer from "information overload".

PATTERN MATCHING OVERCOMES THESE LIMITATIONS IN ACCORDANCE WITH A PREFERRED EMBODIMENT

Here's how the pattern matching system can address each of the corresponding issues above in accordance with a preferred embodiment.

1. By doing pattern matching, we match up only parts of the meeting text that we want and extract those parts.
2. By performing pattern matching on the meeting body and extracting only the parts from the meeting body that we want. Our meeting body will not go to complete waste then.
3. Pattern matching is based on a set of templates that we specify, allowing us to identify people names, company names etc from a meeting text.
4. In summary, with pattern matching, we no longer suffer from information overload. Of course, the big problem is how well our pattern matching works. If we rely exclusively on artificial intelligence processing, we do not have a 100% hit rate. We are able to identify about 20% of all company names presented to us.

PATTERNS

A pattern in the context of a preferred embodiment is a template specifying the structure of a phrase we are looking for in a meeting text. The patterns supported by a preferred embodiment are selected because they are templates of phrases which have a high probability of appearing in someone's meeting text. For example, when entering a meeting in a calendar, many would write something such as "Meet with Bob Dutton from Stanford University next Tuesday." A common pattern would then be something like the word "with" followed by a person's name (in this example it is Bob Dutton) followed by the word "from" and ending with an organization's name (in this case, it is Stanford University).

PATTERN MATCHING TERMINOLOGY

The common terminology associated with pattern matching is provided below.

- Pattern: a pattern is a template specifying the structure of a phrase we want to bind the meeting text to. It contains sub units.
- Element: a pattern can contain many sub-units. These subunits are called elements. For example, in the pattern "with $PEOPLE$ from $COMPANY$", "with" "$PEOPLE$" "from" "$COMPANY$" are all elements.
- Placeholder: a placeholder is a special kind of element in which we want to bind a value to.Using the above example, "$PEOPLE$" is a placeholder.
- Indicator: an indicator is another kind of element which we want to find in a meeting text but no value needs to bind to it. There may be often more than one indicator we are looking for in a certain pattern. That is why an indicator is not an "atomic" type.
- Substitute: substitutes are a set of indicators which are all synonyms of each other. Finding any one of them in the input is good.

There are five fields which are identified for each meeting:

- Company ($COMPANY$)
- People ($PEOPLE$)
- Location ($LOCATION$)
- Time ($TIME$)
- Topic ($TOPIC_UPPER$) or ($TOPIC_ALL$)

In parentheses are the placeholders I used in my code as representation of the corresponding meeting fields.

Each placeholder has the following meaning:

- $COMPANY$: binds a string of capitalized words (e.g. Meet with Joe Carter of <Andersen Consulting >)
- $PEOPLE$: binds series of string of two capitalized words potentially connected by "," "and" or "&" (e.g. Meet with <Joe Carter> of Andersen Consulting, Meet with <Joe Carter and Luke Hughes> of Andersen Consulting)
- $LOCATION$: binds a string of capitalized words (e.g. Meet Susan at <Palo Alto Square>)
- $TIME$: binds a string containing the format #:## (e.g. Dinner at <6:30 pm>)
- $TOPIC_UPPER$: binds a string of capitalized words for our topic (e.g. <Stanford Engineering Recruiting> Meeting to talk about new hires).
- $TOPIC_ALL$: binds a string of words without really caring if it's capitalized or not. (e.g. Meet to talk about <ubiquitous computing>)

Here is a table representing all the patterns supported by BF. Each pattern belongs to a pattern group. All patterns within a pattern group share a similar format and they only differ from each other in terms of what indicators are used as substitutes. Note that the patterns which are grayed out are also commented in the code. BF has the capability to support these patterns but we decided that matching these patterns is not essential at this point.

PAT GRP	PAT #	PATTERN	EXAMPLE
1	a	$PEOPLE$ of $COMPANY$	Paul Maritz of Microsoft
	b	$PEOPLE$ from $COMPANY$	Bill Gates, Paul Allen and Paul Maritz from Microsoft
2	a	$TOPIC_UPPER$ meeting	Push Technology Meeting
	b	$TOPIC_UPPER$ mtg	Push Technology Mtg
	c	$TOPIC_UPPER$ demo	Push Technology demo
	d	$TOPIC_UPPER$ interview	Push Technology interview
	e	$TOPIC_UPPER$ presentation	Push Technology presentation
	f	$TOPIC_UPPER$ visit	Push Technology visit
	g	$TOPIC_UPPER$ briefing	Push Technology briefing
	h	$TOPIC_UPPER$ discussion	Push Technology discussion
	i	$TOPIC_UPPER$ workshop	Push Technology workshop

	j	$TOPIC_UPPER$ prep	Push Technology prep
	k	$TOPIC_UPPER$ review	Push Technology review
	l	$TOPIC_UPPER$ lunch	Push Technology lunch
	m	$TOPIC_UPPER$ project	Push Technology project
	n	$TOPIC_UPPER$ projects	Push Technology projects
3	a	$COMPANY$ corporation	Intel Corporation
	b	$COMPANY$ corp.	IBM Corp.
	c	$COMPANY$ systems	Cisco Systems
	d	$COMPANY$ limited	IBM limited
	e	$COMPANY$ ltd	IBM ltd
4	a	about $TOPIC_ALL$	About intelligent agents technology
	b	discuss $TOPIC_ALL$	Discuss intelligent agents technology
	c	show $TOPIC_ALL$	Show the client our intelligent agents technology
	d	re: $TOPIC_ALL$	re: intelligent agents technology
	e	review $TOPIC_ALL$	Review intelligent agents technology
	f	agenda	The agenda is as follows: --clean up --clean up --clean up
	g	agenda: $TOPIC_ALL$	Agenda: --demo client intelligent agents technology. --demo ecommerce.
5	a	w/$PEOPLE$ of $COMPANY$	Meet w/Joe Carter of Andersen Consulting
	b	w/$PEOPLE$ from $COMPANY$	Meet w/Joe Carter from Andersen Consulting
6	a	w/$COMPANY$ per $PEOPLE$	Talk w/Intel per Jason Foster
7	a	At $TIME$	at 3:00pm
	b	Around $TIME$	Around 3:00 pm
8	a	At $LOCATION$	At LuLu's resturant
	b	In $LOCATION$	in Santa Clara
9	a	Per $PEOPLE$	per Susan Butler
10	a	call w/$PEOPLE$	Conf call w/John Smith
	B	call with $PEOPLE$	Conf call with John Smith

11	A	prep for $TOPIC_ALL$	Prep for London meeting
	B	preparation for $TOPIC_ALL$	Preparation for London meeting

Figure 4 is a detailed flowchart of pattern matching in accordance with a preferred embodiment. Processing commences at function block 400 where the main program invokes the pattern matching application and passes control to function block 410 to commence the pattern match processing. Then, at function block 420, the wrapper function loops through to process each pattern which includes determining if a part of the text string can be bound to a pattern as shown in function block 430. Then, at function block 440, various placeholders are bound to values if they exist, and in function block 441, a list of names separated by punctuation are bound, and at function block 442 a full name is processed by finding two capitalized words as a full name and grabbing the next letter after a space after a word to determine if it is capitalized. Then, at function block 443, time is parsed out of the string in an appropriate manner and the next word after a blank space in function block 444. Then, at function block 445, the continuous phrases of capitalized words such as company, topic or location are bound and in function block 446, the next word after the blank is obtained for further processing in accordance with a preferred embodiment. Following the match meeting field processing, function block 450 is utilized to loacte an indicator which is the head of a pattern, the next word after the blank is obtained as shown in function block 452 and the word is checked to determine if the word is an indicator as shown in function block 454. Then, at function block 460, the string is parsed to locate an indicator which is not at the end of the pattern and the next word after unnecessary white space such as that following a line feed or a carriage return is processed as shown in function block 462 and the word is analyzed to determine if it is an indicator as shown in function block 464. Then, in function block 470, the temporary record is reset to the null set to prepare it for processing the next string and at function block 480, the meeting record is updated and at function block 482 a check is performed to determine if an entry is already made to the meeting record before parsing the meeting record again.

USING THE IDENTIFIED MEETING FIELDS

Now that we have identified fields within the meeting text which we consider important, there are quite a few things we can do with it. One of the most important applications of pattern matching is of course to improve the query we construct which eventually gets submitted to Alta Vista and News Page. There are also a lot of other options and enhancements which exploit the results of pattern matching that we can add to BF. These other options will be described in the next section. The goal of this section is to give the reader a good sense of how the results obtained from pattern matching can be used to help us obtain better search results.

Figure 5 is a flowchart of the detailed processing for preparing a query and obtaining information from the Internet in accordance with a preferred embodiment. Processing commences at function block 500 and immediately flows to function block 510 to process the wrapper functionality to prepare for an Internet search utilizing a web search engine. If the search is to utilize the Alta Vista search engine, then at function block 530, the system takes information from the meeting record and forms a query in function blocks 540 to 560 for submittal to the search engine. If the search is to utilize the NewsPage search engine, then at function block 520, the system takes information from the meeting record and forms a query in function blocks 521 to 528.

Alta Vista Search Engine

The strength of the Alta Vista search engine is that it provides enhanced flexibility. Using its advance query method, one can construct all sorts of Boolean queries and rank the search however you want. However, one of the biggest drawbacks with Alta Vista is that it is not very good at handling a large query and is likely to give back irrelevant results. If we can identify the topic and the company within a meeting text, we can form a pretty short but comprehensive query which will hopefully yield better results. We also want to focus on the topics found. It may not be of much merit to the user to find out info about a company especially if the user already knows the company well and has had numerous meetings with them. It's the topics they want to research on.

News Page Search Engine

The strength of the News Page search engine is that it does a great job searching for the most recent news if you are able to give it a valid company name. Therefore when we submit a query to the news page web site, we send whatever company name we can identify and only if we cannot find one do we use the topics found to form a query. If neither one is found, then no search is performed. The algorithmn utilized to form the query to submit to Alta Vista is illustrated in Figure 7. The algorithmn that we will use to form the query to submit to News Page is illustrated in Figure 8.

The following table describes in detail each function in accordance with a preferred embodiment. The order in which functions appear mimics the process flow as closely as possible. When there are situations in which a function is called several times, this function will be listed after the first function which calls it and its description is not duplicated after every subsequent function which calls it.

Procedure Name	Type	Called By	Description
Main (BF.Main)	Public Sub	None	This is the main function where the program first launches. It initializes BF with the appropriate parameters(e.g. Internet time-out, stoplist...) and calls GoBF to launch the main part of the program.
ProcessCommandLine (BF.Main)	Private Sub	Main	This function parses the command line. It assumes that the delimiter indicating the beginning of input from Munin is stored in the constant CMD_SEPARATOR.
CreateStopList (BF.Main)	Private Function	Main	This function sets up a stop list for future use to parse out unwanted words from the meeting text. There are commas on each side of each word to enable straight checking.
CreatePatterns (BF.PatternMatch	Public Sub	Main	This procedure is called once when BF is first initialized to create all the

Procedure Name	Type	Called By	Description
)			potential patterns that portions of the meeting text can bind to. A pattern can contain however many elements as needed. There are two types of elements. The first type of elements are indicators. These are real words which delimit the potential of a meeting field (eg company) to follow. Most of these indicators are stop words as expected because stop words are words usually common to all meeting text so it makes sense they form patterns. The second type of elements are special strings which represent placeholders. A placeholder is always in the form of $*$ where * can be either PEOPLE, COMPANY, TOPIC_UPPER, TIME,LOCATION or TOPIC_ALL. A pattern can begin with either one of the two types of elements and can be however long, involving however any number/type of elements. This procedure dynamically creates a new pattern record for each pattern in the table and it also dynamically creates new tAPatternElements for each element within a pattern. In addition, there is the concept of being able to substitute indicators within a pattern. For example, the pattern $PEOPLE$ of $COMPANY$ is similar to the pattern $PEOPLE$ from $COMPANY$. "from" is a substitute for "of" . Our structure should be able to express such a need for substitution.

Procedure Name	Type	Called By	Description
GoBF (BF.Main)	Public Sub	Main	This is a wrapper procedurer that calls both the parsing and the searching subroutines of the BF. It is also responsible for sending data back to Munin.
ParseMeetingText (BF.Parse)	Public Function	GoBackGroundFinder	This function takes the initial meeting text and identifies the userID of the record as well as other parts of the meeting text including the title, body, participant list, location and time. In addition, we call a helper function ProcessStopList to eliminate all the unwanted words from the original meeting title and meeting body so that only keywords are left. The information parsed out is stored in the MeetingRecord structure. Note that this function does no error checking and for the most time assumes that the meeting text string is correctly formatted by Munin. The important variable is thisMeeting Record is the temp holder for all info regarding current meeting. It's eventually returned to caller.
FormatDelimitation (BF.Parse)	Private	ParseMeetingText, DetermineNumWords, GetAWordFromString	There are 4 ways in which the delimiters can be placed. We take care of all these cases by reducing them down to Case 4 in which there are no delimiters around but only between fields in a string(e.g. A::B::C)
DetermineNumWords (BF.Parse)	Public Function	ParseMeeting Text, ProcessStop List	This functions determines how many words there are in a string (stInEvalString) The function assumes that each word is separated by a designated separator as specified in stSeparator. The return type is an

Procedure Name	Type	Called By	Description
			integer that indicates how many words have been found assuming each word in the string is separated by stSeparator. This function is always used along with GetAWordFromString and should be called before calling GetAWordFrom String.
GetAWordFromString (BF.Parse)	Public Function	ParseMeeting Text, ProcessStop List	This function extracts the ith word of the string(stInEvalString) assuming that each word in the string is separated by a designated separator contained in the variable stSeparator. In most cases, use this function with DetermineNumWords. The function returns the wanted word. This function checks to make sure that iInWordNum is within bounds so that i is not greater than the total number of words in string or less than/equal to zero. If it is out of bounds, we return empty string to indicate we can't get anything. We try to make sure this doesn't happen by calling DetermineNumWords first.
ParseAndCleanPhrase (BF.Parse)	Private Function	ParseMeetingText	This function first grabs the word and send it to CleanWord in order strip the stuff that nobody wants. There are things in parseWord that will kill the word, so we will need a method of looping through the body and rejecting words without killing the whole function i guess keep CleanWord and check a return value ok, now I have a word so I need to send it down the parse chain. This chain goes

Procedure Name	Type	Called By	Description
			ParseCleanPhrase -> CleanWord -> EvaluateWord. If the word gets through the entire chain without being killed, it will be added at the end to our keyword string. first would be the function that checks for "/" as a delimiter and extracts the parts of that. This I will call "StitchFace" (Denise is more normal and calls it GetAWordFromString) if this finds words, then each of these will be sent, in turn, down the chain. If these get through the entire chain without being added or killed then they will be added rather than tossed.
FindMin (BF.Parse)	Private Function	ParseAndCleanPhrase	This function takes in 6 input values and evaluates to see what the minimum non zero value is. It first creates an array as a holder so that we can sort the five input values in ascending order. Thus the minimum value will be the first non zero value element of the array. If we go through entire array without finding a non zero value, we know that there is an error and we exit the function.
CleanWord (BF.Parse)	Private Function	ParseAndCleanPhrase	This function tries to clean up a word in a meeting text. It first of all determines if the string is of a valid length. It then passes it through a series of tests to see it is clean and when needed, it will edit the word and strip unnecessary characters off of it. Such tests includes getting rid of file extensions,

Procedure Name	Type	Called By	Description
			non chars, numbers etc.
EvaluateWord (BF.Parse)	Private Function	ParseAndCleanPhrase	This function tests to see if this word is in the stop list so it can determine whether to eliminate the word from the original meeting text. If a word is not in the stoplist, it should stay around as a keyword and this function exits beautifully with no errors. However, if the words is a stopword, an error must be returned. We must properly delimit the input test string so we don't accidentally retrieve sub strings.
GoPatternMatch (BF.PatternMatch)	Public Sub	GoBF	This procedure is called when our QueryMethod is set to complex query meaning we do want to do all the pattern matching stuff.It 's a simple wrapper function which initializes some arrays and then invokes pattern matching on the title and the body.
MatchPatterns (BF.PatternMatch)	Public Sub	GoPattern Match	This procedure loops through every pattern in the pattern table and tries to identify different fields within a meeting text specified by sInEvalString. For debugging purposes it also tries to tabulate how many times a certain pattern was triggered and stores it in gTabulateMatches to see whichp pattern fired the most. gTabulateMatches is stored as a global because we want to be able to run a batch file of 40 or 50 test strings and still be able to know how often a pattern was triggered.
MatchAPattern (BF.PatternMatch)	Private Function	MatchPatterns	This function goes through each element in the current pattern. It first evaluates to determine whether element is a placeholder or an

Procedure Name	Type	Called By	Description
			indicator. If it is a placeholder, then it will try to bind the placeholder with some value. If it is an indicator, then we try to locate it. There is a trick however. Depending on whether we are at current element is the head of the pattern or not we want to take different actions. If we are at the head, we want to look for the indicator or the placeholder. If we can't find it, then we know that the current pattern doesn't exist and we quit. However, if it is not the head, then we continue looking, because there may still be a head somewhere. We retry in this case.
MatchMeetingField (BF.PatternMatch)	Private Function	MatchAPattern	This function uses a big switch statement to first determine what kind of placeholder we are talking about and depending on what type of placeholder, we have specific requirements and different binding criteria as specified in the subsequent functions called such as BindNames, BindTime etc. If binding is successful we add it to our guessing record.
BindNames (BF.PatternMatch)	Private Function	MatchMeetingField	In this function, we try to match names to the corresponding placeholder $PEOPLE$. Names are defined as any consecutive two words which are capitalized. We also what to retrieve a series of names which are connected by and , or & so we look until we don't see any of these 3 separators anymore. Note that we don 't want to bind single word names because it is

Procedure Name	Type	Called By	Description
			probably too general anyway so we don't want to produce broad but irrelevant results. This function calls BindAFullName which binds one name so in a since BindNames collects all the results from BindAFullName
BindAFullName (BF.PatternMatch)	Private Function	BindNames	This function tries to bind a full name. If the $PEOPLE$ placeholder is not the head of the pattern, we know that it has to come right at the beginning of the test string because we 've been deleting stuff off the head of the string all along. If it is the head, we search until we find something that looks like a full name. If we can't find it, then there's no such pattern in the text entirely and we quit entirely from this pattern. This should eventually return us to the next pattern in MatchPatterns.
GetNextWordAfterWhiteSpace (BF.PatternMatch)	Private Function	BindAFull Name, BindTime, BindCompanyTopicLoc	This function grabs the next word in a test string. It looks for the next word after white spaces, @ or /. The word is defined to end when we encounter another one of these white spaces or separators.
BindTime (BF.PatternMatch)	Private Function	MatchMeetingField	Get the immediate next word and see if it looks like a time pattern. If so we've found a time and so we want to add it to the record. We probably should add more time patterns. But people don't seem to like to enter the time in their titles these days especially since we now have tools like OutLook.
BindCompanyTopicLoc	Private Function	MatchMeetingField	This function finds a continuous capitalized string and binds it to

Procedure Name	Type	Called By	Description
(BF.PatternMatch)			stMatch which is passed by reference from MatchMeetingField. A continous capitalized string is a sequence of capitalized words which are not interrupted by things like , . etc. There's probably more stuff we can add to the list of interruptions.
LocatePatternHead (BF.PatternMatch)	Private Function	MatchAPattern	This function tries to locate an element which is an indicator. Note that this indicator SHOULD BE AT THE HEAD of the pattern otherwise it would have gone to the function LocateIndicator instead. Therefore, we keep on grabbing the next word until either there's no word for us to grab (quit) or if we find one of the indicators we are looking for.
ContainInArray (BF.PatternMatch)	Private Function	LocatePattern Head, LocateIndicator	' This function is really simple. It loops through all the elements in the array ' to find a matching string.
LocateIndicator (BF.PatternMatch)	Private Function	MatchAPattern	This function tries to locate an element which is an indicator. Note that this indicator is NOT at the head of the pattern otherwise it would have gone to LocatePatternHead instead. Because of this, if our pattern is to be satisfied, the next word we grab HAS to be the indicator or else we would have failed. Thus we only grab one word, test to see if it is a valid indicator and then return result.
InitializeGuesses Record (BF.PatternMatch)	Private Sub	MatchAPattern	This function reinitializes our temporary test structure because we have already transfered the info to the permanent structure, we can reinitialize it so they each have one

Procedure Name	Type	Called By	Description
			element
AddToMeetingRecord (BF.PatternMatch)	Private Sub	MatchAPattern	This function is only called when we know that the information stored in tInCurrGuesses is valid meaning that it represents legitamate guesses of meeting fields ready to be stored in the permanent record,tInMeetingRecord. We check to make sure that we do not store duplicates and we also what to clean up what we want to store so that there's no cluttered crap such as punctuations, etc. The reason why we don't clean up until now is to save time. We don't waste resources calling ParseAndCleanPhrase until we know for sure that we are going to add it permanently.
NoDuplicateEntry (BF.PatternMatch)	Private Function	AddToMeetingRecord	This function loops through each element in the array to make sure that the test string aString is not the same as any of the strings already stored in the array. Slightly different from ContainInArray.
SearchAltaVista (BF.Search)	Public Function	GoBackGroundFinder	This function prepares a query to be submited to AltaVista Search engine. It submits it and then parses the returning result in the appropriate format containing the title, URL and body/summary of each story retrieved. The number of stories retrieved is specified by the constant NUM_AV_STORIES. Important variables include stURLAltaVista used to store query to submit stResultHTML used to store html from page specified by stURLAltaVista.

Procedure Name	Type	Called By	Description
ConstructAltaVistaURL (BF.Search)	Private Function	SearchAltaVista	This function constructs the URL string for the alta vista search engine using the advanced query search mode. It includes the keywords to be used, the language and how we want to rank the search. Depending on whether we want to use the results of our pattern matching unit, we construct our query differently.
ConstructSimpleKeyWord (BF.Search)	Private Function	ConstructAltaVistaURl, ConstructNewsPageURL	This function marches down the list of keywords stored in the stTitleKW or stBodyKW fields of the input meeting record and links them up into one string with each keyword separated by a connector as determined by the input variable stInConnector. Returns this newly constructed string.
ConstructComplexAVKeyWord (BF.Search)	Private Function	ConstructAltaVistaURL	This function constructs the keywords to be send to the AltaVista site. Unlike ConstructSimpleKeyWord which simply takes all the keywords from the title to form the query, this function will look at the results of BF 's pattern matching process and see if we are able to identify any specific company names or topics for constructing the queries. Query will include company and topic identified and default to simple query if we cannot identify either company or topic.
JoinWithConnectors (BF.Search)	Private Function	ConstructComplexAVKey Word, ConstructComplexNPKey Word, RefineWith	This function simply replaces the spacesbetween the words within the string with a connector which is specified by the input.

Procedure Name	Type	Called By	Description
		Rank	
RefineWithDate (NOT CALLED AT THE MOMENT) (BF.Search)	Private Function	ConstructAltaVistaURL	This function constructs the date portion of the alta vista query and returns this portion of the URL as a string. It makes sure that alta vista searches for articles within the past PAST_NDAYS.
RefineWithRank (BF.Search)	Private Function	ConstructAltaVistaURL	This function constructs the string needed to passed to Altavista in order to rank an advanced query search. If we are constructing the simple query we will take in all the keywords from the title. For the complex query, we will take in words from company and topic, much the same way we formed the query in ConstructComplexAVKeyWord.
IdentifyBlock (BF.Parse)	Public Function	SearchAltaVista, SearchNewsPage	This function extracts the block within a string marked by the beginning and the ending tag given as inputs starting at a certain location(iStart). The block retrieved does not include the tags themselves. If the block cannot be identified with the specified delimiters, we return unsuccessful through the parameter iReturnSuccess passed to use by reference. The return type is the block retrieved.
IsOpenURLError (BF.Error)	Public Function	SearchAltaVista,SearchNewsPage	This function determines whether the error encountered is that of a timeout error. It restores the mouse to default arrow and then returns true if it is a time out or false otherwise.
SearchNewsPage (BF.Search)	Public Function	GoBackGroundFinder	This function prepares a query to be submited to NewsPage Search engine. It submits it and then parses the returning result in the appropriate

Procedure Name	Type	Called By	Description
			format containing the title, URL and body/summary of each story retrieved. The number of stories retrieved is specified by the constant UM_NP_STORIES
ConstructNewsPageURL (BF.Search)	Private Function	SearchNewsPage	This function constructs the URL to send to the NewsPage site. It uses the information contained in the input meeting record to determine what keywords to use. Also depending whether we want simple or complex query, we call diffent functions to form strings.
ConstructComplexNPKeyWord (BF.Search)	Private Function	ConstructNewsPageURL	This function constructs the keywords to be send to the NewsPage site. UnlikeConstructKeyWordString which simply takes all the keywords from the title to form the query, this function will look at the results of BF 's pattern matching process and see if we are able to identify any specific company names or topics for constructing the queries. Since newspage works best when we have a company name, we 'll use only the company name and only if there is no company will we use topic.
ConstructOverall Result (BF.Main)	Private Function	GoBackGroundFinder	This function takes in as input an array of strings (stInStories) and a MeetingRecord which stores the information for the current meeting. Each element in the array stores the stories retrieved from each information source. The function simply constructs the appropriate output to send to Munin including a return message type to let Munin know that it is the BF

Procedure Name	Type	Called By	Description
			responding and also the original user_id and meeting title so Munin knows which meeting BF is talking about.
ConnectAndTransferTo Munin (BF.Main)	Public Sub	GoBackGroundFinder	This function allows Background Finder to connect to Munin and eventually transport information to Munin. We will be using the UDP protocol instead of the TCP protocol so we have to set up the remote host and port correctly. We use a global string to store gResult Overall because although it is unecessary with UDP, it is needed with TCP and if we ever switch back don't want to change code.
DisconnectFrom MuninAnd Quit (BF.Main)	Public Sub		

Figure 6 is a flowchart of the actual code utilized to prepare and submit searches to the Alta Vista and Newspage search engines in accordance with a preferred embodiment. Processing commences at function block **610** where a command line is utilized to update a calendar entry with specific calendar information. The message is next posted in accordance with function block **620** and a meeting record is created to store the current meeting information in accordance with function block **630**. Then, in function block **640** the query is submitted to the Alta Vista search engine and in function block **650**, the query is submitted to the Newspage search engine. When a message is returned from the search engine, it is stored in a results data structure as shown in function block **660** and the information is processed and stored in summary form in a file for use in preparation for the meeting as detailed in function block **670**.

Figure 7 provides more detail on creating the query in accordance with a preferred embodiment. Processing commences at function block **710** where the meeting record is parsed to obtain potential companies, people, topics, location and a time. Then, in function block **720**, at least one topic is identified and in function block **720**, at least one company name is identified and finally in function block **740**, a decision is made on what material to transmit to the file for ultimate consumption by the user.

Figure 8 is a variation on the query theme presented in Figure 7. A meeting record is parsed in function block **800**, a company is identified in function block **820**, a topic is identified in function block **830** and finally in function block **840** the topic and or the company is utilized in formulating the query.

Alternative embodiments for adding various specific features for specific user requirements are discussed below.

Enhance Target Rate for Pattern Matching

To increase BF's performance, more patterns/pattern groups are added to the procedure "CreatePatterns." The existing code for declaring patterns can be used as a template for future patterns. Because everything is stored as dynamic arrays, it is convenient to reuse code by cutting and pasting. The functions BindName, BindTime, BindCompanyLocTopic which are responsible for associating a value with a placeholder can be enhanced. The enhancement is realized by increasing the set of criteria for binding a certain meeting field in order to increase the number of binding values. For example, BindTime currently accepts and binds all values in the form of ##:## or #:##. To increase the times we can bind, we may want BindTime to also accept the numbers 1 to 12 followed by the more aesthetic time terminology "o'clock." Vocabulary based recognition algorithms and assigning an accuracy rate to each guess BF makes allowing only guesses which meet a certain threshold to be valid.

Depending on what location the system identifies through pattern matching or alternatively depending on what location the user indicates as the meeting place, a system in accordance with a preferred embodiment suggests a plurality of fine restaurants whenever it detects the words lunch/dinner/breakfast. We can also use a site like company finder to confirm what we got is indeed a company name or if there is no company name that pattern matching can identify, we can use a company finder web site as a "dictionary" for us to determine whether certain capitalized words represent a company name. We can even display stock prices and breaking news for a company that we have identified.

Wireless Bargain Identification in Accordance With A Preferred Embodiment

Figure 9 is a flow diagram that depicts the hardware and logical flow of control for a device and a software system designed to allow Web-based comparison shopping in conventional, physical, non-Web retail environments. A wireless phone or similar hand-held wireless device **920** with Internet Protocol capability is combined with a miniature barcode reader **910** (installed either inside the phone or on a short cable) and used to scan the Universal Product Code (UPC) bar code on a book or other product **900**. The wireless device **920** transmits the bar code via an antennae **930** to the Pocket BargainFinder Service Module (running on a Web server) **940**, which converts it to (in the case of books) its International Standard Book Number or (in the case of other products) whatever identifier is appropriate. The Service Module then contacts the appropriate third-party Web site(s) to find price, shipping and availability information on the product from various Web suppliers **950**. This information is formatted and displayed on the hand-held device's screen. The IP wireless phone or other hand held device **920** utilizes a wireless modem such as a Ricochet SE Wireless Modem from Metricom. Utilizing this device, a user can hang out in a coffee shop with a portable computer perched on a rickety little table, with a latte sloshing dangerously close to the keyboard, and access the Internet at speeds rivaling direct connect via a telephone line.

The 8-ounce Ricochet SE Wireless Modem is about as large as a pack of cigarettes and setup is extremely simple, simply attach the modem to the back of your portable's screen with the included piece of Velcro, plug the cable into the serial port, flip up the stubby antenna, and transmit. Software setup is equally easy: a straightforward installer adds the Ricochet modem drivers and places the connection icon on your desktop. The functional aspects of the modem are identical to that of a traditional telephone modem.

Of course, wireless performance isn't nearly as reliable as a traditional dial-up phone connection. We were able to get strong connections in several San Francisco locations as long as we stayed near the windows. But inside CNET's all-brick headquarters, the Ricochet couldn't connect at all. When you do get online, performance of up to 28.8 kbps is available with graceful degradation to slower speeds. But even the slower speeds didn't disappoint. Compared to the alternative--connecting via a cellular modem--the Ricochet is much faster, more reliable, and less expensive to use. Naturally, the SE Wireless is battery powered. The modem has continuous battery life of up to 12 hours. And in accordance with a preferred embodiment, we ran down our portable computer's dual cells before the Ricochet started to fade.

Thus, utilizing the wireless modem, a user may utilize the web server software **940** to identify the right product **950** and then use an appropriate device's key(s) to select a supplier and place an order in accordance with a preferred embodiment. The BargainFinder Service Module then consummates the order with the appropriate third-party Web supplier **960**.

mySite! Personal Web Site & Intentions Value Network Prototype

mySite! is a high-impact, Internet-based application in accordance with a preferred embodiment that is focused on the theme of delivering services and providing a personalized experience for each customer via a personal web site in a buyer-centric world. The services are intuitively organized around satisfying customer intentions - fundamental life needs or objectives that require

extensive planning decisions, and coordination across several dimensions, such as financial planning, healthcare, personal and professional development, family life, and other concerns. Each member owns and maintains his own profile, enabling him to create and browse content in the system targeted specifically at him. From the time a demand for products or services is entered, to the completion of payment, intelligent agents are utilized to conduct research, execute transactions and provide advice. By using advanced profiling and filtering, the intelligent agents learn about the user, improving the services they deliver. Customer intentions include Managing Daily Logistics (e.g., email, calendar, contacts, to-do list, bill payment, shopping, and travel planning); and Moving to a New Community (e.g., finding a place to live, moving household possessions, getting travel and shipping insurance coverage, notifying business and personal contacts, learning about the new community). From a consumer standpoint, mySite! provides a central location where a user can access relevant products and services and accomplish daily tasks with ultimate ease and convenience.

From a business standpoint, mySite! represents a value-added and innovative way to effectively attract, service, and retain customers. Intention value networks allow a user to enter through a personalized site and, and with the assistance of a learning, intelligent agent, seamlessly interact with network participants. An intention value network in accordance with a preferred embodiment provides superior value. It provides twenty four hour a day, seven days a week access to customized information, advice and products. The information is personalized so that each member views content that is highly customized to assure relevance to the required target user.

Egocentric Interface

An Egocentric Interface is a user interface crafted to satisfy a particular user's needs, preferences and current context. It utilizes the user's personal information that is stored in a central profile database to customize the interface. The user can set security permissions on and preferences for interface elements and content. The content integrated into the Egocentric Interface is customized with related information about the user. When displaying content, the Egocentric Interface will include the relationship between that content and the user in a way that demonstrates how the content relates to the user. For instance, when displaying information about an upcoming ski trip the user has signed up for, the interface will include information about events from the user's personal calendar and contact list, such as other people who will be in the area during the ski trip. This serves to put the new piece of information into a context familiar to the individual user.

Figure **10A** describes the **Intention Value Network Architecture** implementation for the World Wide Web. For simplification purposes, this diagram ignores the complexity pertaining to security, scalability and privacy. The customer can access the Intention Value Network with any Internet web browser **1010**, such as Netscape Navigator or Microsoft Internet Explorer, running on a personal computer connected to the Internet or a Personal Digital Assistant with wireless capability. See Figure **17** for a more detailed description of the multiple methods for accessing an Intention Value Network. The customer accesses the Intention Value Network through the unique name or IP address associated with the Integrator's Web Server **1020**. The Integrator creates the Intention Value Network using a combination of resources, such as the Intention Database **1030**, the Content Database **1040**, the Supplier Profile Database **1050**, and the Customer Profile Database **1060**.

The Intention Database **1030** stores all of the information about the structure of the intention and the types of products and services needed to fulfill the intention. Information in this database includes intention steps, areas of interest, layout templates

and personalization templates. The Content Database **1040** stores all of the information related to the intention, such as advice, referral information, personalized content, satisfaction ratings, product ratings and progress reports.

The Supplier Profile Database **1050** contains information about the product and service providers integrated into the intention. The information contained in this database provides a link between the intention framework and the suppliers. It includes product lists, features and descriptions, and addresses of the suppliers' product web sites. The Customer Profile Database **1060** contains personal information about the customers, such as name, address, social security number and credit card information, personal preferences, behavioral information, history, and web site layout preferences. The Supplier's Web Server **1070** provides access to all of the supplier's databases necessary to provide information and transactional support to the customer.

The Product Information Database **1080** stores all product-related information, such as features, availability and pricing. The Product Order Database **1090** stores all customer orders. The interface to this database may be through an Enterprise Resource Planning application offered by SAP, Baan, Oracle or others, or it may be accessible directly through the Supplier's Web Server or application server. The Customer Information Database **1091** stores all of the customer information that the supplier needs to complete a transaction or maintain customer records.

Figure **10B** is a flowchart providing the logic utilized to create a web page within the Egocentric Interface. The environment assumes a web server and a web browser connected through a TCP/IP network, such as over the public Internet or a private Intranet. Possible web servers could include Microsoft Internet Information Server, Netscape Enterprise Server or Apache. Possible web browsers include Microsoft Internet Explorer or Netscape Navigator. The client (i.e. web browser) makes a request **1001** to the server (i.e. web server) for a particular web page. This is usually accomplished by a user clicking on a button or a link within a web page. The web server gets the layout and content preferences **1002** for that particular user, with the request to the database keyed off of a unique user id stored in the client (i.e. web browser) and the User profile database **1003**. The web server then retrieves the content **1004** for the page that has been requested from the content database **1005**. The relevant user-centric content, such as calendar, email, contact list, and task list items are then retrieved **1006**. (See **Figure 11** for a more detailed description of this process.) The query to the database utilizes the user content preferences stored as part of the user profile in the User profile database **1003** to filter the content that is returned. The content that is returned is then formatted into a web page **1007** according to the layout preferences defined in the user profile. The web page is then returned to the client and displayed to the user **1008**.

Figure 11 describes the process of retrieving user-centric content to add to a web page. This process describes **1006** in **Figure 10B** in a more detailed fashion. It assumes that the server already has obtained the user profile and the existing content that is going to be integrated into this page. The server parses **1110** the filtered content, looking for instances of events, contact names and email addresses. If any of these are found, they are tagged and stored in a temporary holding space. Then, the server tries to find any user-centric content **1120** stored in various databases. This involves matching the tagged items in the temporary storage space with calendar items **1130** in the Calendar Database **1140**; email items **1115** in the Email Database **1114**; contact items **1117** in the Contact Database **1168**; task list items **1119** in the Task List Database **1118**; and news items **1121** in the News Database **1120**. After retrieving any relevant user-centric content, it is compiled together and returned **1122**.

User Persona

The system allows the user to create a number of different personas that aggregate profile information into sets that are useful in different contexts. A user may create one persona when making purchases for his home. This persona may contain his home address and may indicate that this user is looking to find a good bargain when shopping. The same user may create a second persona that can be used when he is in a work context. This persona may store the user's work address and may indicate that the user prefers certain vendors or works for a certain company that has a discount program in place. When shopping for work-related items, the user may use this persona. A persona may also contain rules and restrictions. For instance, the work persona may restrict the user to making airline reservations with only one travel agent and utilizing booking rules set up by his employer.

Figure 12 describes the relationship between a user, his multiple personas and his multiple profiles. At the User Level is the User Profile **1200**. This profile describes the user and his account information. There is one unique record in the database for each user who has an account. Attached to each user are multiple Personas **1220, 1230 & 1240**. These Personas are used to group multiple Profiles into useful contexts. For instance, consider a user who lives in San Francisco and works in Palo Alto, but has a mountain cabin in Lake Tahoe. He has three different contexts in which he might be accessing his site. One context is work-related. The other two are home-life related, but in different locations. The user can create a Persona for Work **1220**, a Persona for Home **1230**, and a Persona for his cabin home **1240**. Each Persona references a different General Profile **1250, 1260** and **1270** which contains the address for that location. Hence, there are three General Profiles. Each Persona also references one of two Travel Profiles. The user maintains a Work Travel Profile **1280** that contains all of the business rules related to booking tickets and making reservations. This Profile may specify, for instance, that this person only travels in Business or First Class and his preferred airline is United Airlines. The Work Persona references this Work Travel Profile. The user may also maintain a Home Travel Profile **1290** that specifies that he prefers to travel in coach and wants to find non-refundable fairs, since they are generally cheaper. Both the Persona for Home and the Persona for the cabin home point to the Home Travel Profile.

Figure **13** describes the data model that supports the Persona concept. The user table **1310** contains a record for each user who has an account in the system. This table contains a username and a password **1320** as well as a unique identifier. Each user can have multiple Personas **1330**, which act as containers for more specialized structures called Profiles **1340**. Profiles contain the detailed personal information in Profile Field **1350** records. Attached to each Profile are sets of Profile Restriction **1360** records. These each contain a Name **1370** and a Rule **1380**, which define the restriction. The Rule is in the form of a pattern like (if x then y), which allows the Rule to be restricted to certain uses. An example Profile Restriction would be the rule that dictates that the user cannot book a flight on a certain airline contained in the list. This Profile Restriction could be contained in the "Travel" Profile of the "Work" Persona set up by the user's employer, for instance. Each Profile Field also contains a set of Permissions **1390** that are contained in that record. These permissions dictate who has what access rights to that particular Profile Field's information.

Intention-Centric Interface

Satisfying Customer Intentions, such as Planning for Retirement or Relocating requires a specialized interface. Customer Intentions require extensive planning and coordination across many areas, ranging from financial security, housing and

transportation to healthcare, personal and professional development, and entertainment, among others. Satisfying Intentions requires a network of complementary businesses, working across industries, to help meet consumers' needs.

An Intention-Centric Interface is a user interface designed to help the user manage personal Intentions. At any given point, the interface content is customized to show only content that relates to that particular Intention. The Intention-Centric Interface allows the user to manage the process of satisfying that particular Intention. This involves a series of discrete steps and a set of content areas the user can access. At any point, the user can also switch the interface to manage a different Intention, and this act will change the content of the interface to include only that content which is relevant to the satisfaction of the newly selected Intention.

Figure **14** provides a detailed description of the data model needed to support an Intention-Centric Interface. Each User Persona **1410** (see Figure **13** for a more detailed description of the Persona data model.) has any number of active User Intentions **1420**. Each active User Intention is given a Nickname **1430**, which is the display name the user sees on the screen. Each active User Intention also contains a number of Data Fields **1440**, which contain any user data collected throughout the interaction with the user. For instance, if the user had filled out a form on the screen and one of the fields was Social Security Number, the corresponding Data Field would contain Name = "SSN" **1450**, Value = "999-99-9999" **1460**. Each User Intention also keeps track of Intention Step **1470** completion status. The Completion **1480** field indicates whether the user has completed the step. Every User Intention is a user-specific version of a Generic Intention **1490**, which is the default model for that Intention for all users. The Generic Intention is customized through Custom Rules **1411** and **1412** that are attached to the sub-steps in the Intention. These Custom Rules are patterns describing how the system will customize the Intention for each individual user using the individual user's profile information.

Statistical Agent

An agent keeps track of key statistics for each user. These statistics are used in a manner similar to the Tamagochi virtual reality pet toy to encourage certain behaviors from the user. The statistics that are recorded are frequency of login, frequency of rating of content such as news articles, and activity of agents, measured by the number of tasks which it performs in a certain period. This information is used by the system to emotionally appeal to the user to encourage certain behaviors.

Figure **15** describes the process for generating the page that displays the agent's current statistics. When the user requests the agent statistics page **1510** with the client browser, the server retrieves the users' statistics **1520** from the users' profile database **1530**. The server then performs the mathematical calculations necessary to create a normalized set of statistics **1540**. The server then retrieves the formulas **1550** from the content database **1560** that will be used to calculate the user-centric statistics. Graphs are then generated **1570** using the generic formulas and that user's statistics. These graphs are inserted into a template to create the statistics page **1580**. This page is then returned to the user **1590**.

Personalized Product Report Service

The system provide Consumer Report-like service that is customized for each user based on a user profile. The system records and provides ratings from users about product quality and desirability on a number of dimensions. The difference between this system and traditional product quality measurement services is that the ratings that come back to the users are personalized.

This service works by finding the people who have the closest match to the user's profile and have previously rated the product being asked for. Using this algorithm will help to ensure that the product reports sent back to the user only contain statistics from people who are similar to that user.

Figure 16 describes the algorithm for determining the personalized product ratings for a user. When the user requests a product report **1610** for product X, the algorithm retrieves the profiles **1620** from the profile database **1630** (which includes product ratings) of those users who have previously rated that product. Then the system retrieves the default thresholds **1640** for the profile matching algorithm from the content database **1650**. It then maps all of the short list of users along several dimensions specified in the profile matching algorithm **1660**. The top n (specified previously as a threshold variable) nearest neighbors are then determined and a test is performed to decide if they are within distance y (also specified previously as a threshold variable) of the user's profile in the set **1670** using the results from the profile matching algorithm. If they are not within the threshold, then the threshold variables are relaxed **1680**, and the test is run again. This processing is repeated until the test returns true. The product ratings from the smaller set of n nearest neighbors are then used to determine a number of product statistics **1690** along several dimensions. Those statistics are inserted into a product report template **1695** and returned to the user **1697** as a product report.

Personal Profile and Services Ubiquity

This system provides one central storage place for a person's profile. This storage place is a server available through the public Internet, accessible by any device that is connected to the Internet and has appropriate access. Because of the ubiquitous accessibility of the profile, numerous access devices can be used to customize services for the user based on his profile. For example, a merchant's web site can use this profile to provide personalized content to the user. A Personal Digital Assistant (PDA) with Internet access can synchronize the person's calendar, email, contact list, task list and notes on the PDA with the version stored in the Internet site. This enables the person to only have to maintain one version of this data in order to have it available whenever it is needed and in whatever formats it is needed.

Figure 17 presents the detailed logic associated with the many different methods for accessing this centrally stored profile. The profile database **1710** is the central storage place for the users' profile information. The profile gateway server **1720** receives all requests for profile information, whether from the user himself or merchants trying to provide a service to the user. The profile gateway server is responsible for ensuring that information is only given out when the profile owner specifically grants permission. Any device that can access the public Internet **1730** over TCP/IP (a standard network communications protocol) is able to request information from the profile database via intelligent HTTP requests. Consumers will be able to gain access to services from devices such as their televisions **1740**, mobile phones, Smart Cards, gas meters, water meters, kitchen appliances, security systems, desktop computers, laptops, pocket organizers, PDAs, and their vehicles, among others. Likewise, merchants **1750** will be able to access those profiles (given permission from the consumer who owns each profile), and will be able to offer customized, personalized services to consumers because of this.

One possible use of the ubiquitous profile is for a hotel chain. A consumer can carry a Smart Card that holds a digital certificate uniquely identifying him. This Smart Card's digital certificate has been issued by the system and it recorded his profile information into the profile database. The consumer brings this card into a hotel chain and checks in. The hotel employee swipes the Smart Card and the consumer enters his Pin number, unlocking the digital certificate. The certificate is sent to the

profile gateway server (using a secure transmission protocol) and is authenticated. The hotel is then given access to a certain part of the consumer's profile that he has previously specified. The hotel can then retrieve all of the consumer's billing information as well as preferences for hotel room, etc. The hotel can also access the consumer's movie and dining preferences and offer customized menus for both of them. The hotel can offer to send an email to the consumer's spouse letting him/her know the person checked into the hotel and is safe. All transaction information can be uploaded to the consumer's profile after the hotel checks him in. This will allow partners of the hotel to utilize the information about the consumer that the hotel has gathered (again, given the consumer's permission).

Intention Value Network

In an Intention Value Network, the overall integrator system coordinates the delivery of products and services for a user. The integrator manages a network of approved suppliers providing products and services, both physical and virtual, to a user based on the user's preferences as reflected in the user's profile. The integrator manages the relationship between suppliers and consumers and coordinates the suppliers' fulfillment of consumers' intentions. It does this by providing the consumer with information about products and suppliers and offering objective advice, among other things.

Figure **18** discloses the detailed interaction between a consumer and the integrator involving one supplier. The user accesses a Web Browser **1810** and requests product and pricing information from the integrator. The request is sent from the user's browser to the integrator's Web/Application Server **1820**. The user's preferences and personal information is obtained from an integrator's customer profile database **1830** and returned to the Web/Application server. The requested product information is extracted from the supplier's product database **1840** and customized for the particular customer. The Web/Application server updates the supplier's customer information database **1850** with the inquiry information about the customer. The product and pricing information is then formatted into a Web Page **1860** and returned to the customer's Web Browser.

Summary Agent

A suite of software agents running on the application and web servers are programmed to take care of repetitive or mundane tasks for the user. The agents work according to rules set up by the user and are only allowed to perform tasks explicitly defined by the user. The agents can take care of paying bills for the user, filtering content and emails, and providing a summary view of tasks and agent activity. The user interface for the agent can be modified to suit the particular user.

Figure **19** discloses the logic in accordance with a preferred embodiment processing by an agent to generate a verbal summary for the user. When the user requests the summary page **1900**, the server gets the user's agent preferences **1920**, such as agent type, rules and summary level from the user profile database **1930**. The server gets the content **1940**, such as emails, to do list items, news, and bills, from the content database **1950**. The agent parses all of this content, using the rules stored in the profile database, and summarizes the content **1960**. The content is formatted into a web page **1970** according to a template. The text for the agent's speech is generated **1980**, using the content from the content database **1990** and speech templates stored in the database. This speech text is inserted into the web page **1995** and the page is returned to the user **1997**.

Trusted Third Party

The above scenario requires the web site to maintain a guarantee of privacy of information according to a published policy. This system is the consumer's Trusted Third Party, acting on his behalf in every case, erring on the side of privacy of information,

rather than on the side of stimulation of commerce opportunities. The Trusted Third Party has a set of processes in place that guarantee certain complicity with the stated policy.

"meCommerce"

This word extends the word "eCommerce" to mean "personalized electronic commerce."

Figure **20** illustrates a display login in accordance with a preferred embodiment. The display is implemented as a Microsoft Internet Explorer application with an agent **2000** that guides a user through the process of interacting with the system to customize and personalize various system components to gather information and interact with the user's personal requirements. A user enters a username at **2010** and a password at **2020** and selects a button **2040** to initiate the login procedure. As the logo **2030** suggests, the system transforms electronic commerce into a personalized, so called "me" commerce.

Figure **21** illustrates a managing daily logistics display in accordance with a preferred embodiment. A user is greeted by an animated agent **2100** with a personalized message **2190**. The user can select from various activities based on requirements, including travel **2110**, household chores **2120**, finances **2130** and marketplace activities **2140**. Icons **2142** for routine tasks such as e-mail, calendaring and document preparation are also provided to facilitate rapid navigation from one activity to another. Direct links **2146** are also provided to allow transfer of news and other items of interest. Various profiles can be selected based on where the user is located. For example, work, home or vacation. The profiles can be added **2170** as a user requires a new profile for another location. Various items **2180** of personal information are collected from the user to support various endeavors. Moreover, permissions **2150** are set for items **2180** to assure information is timely and current.

Figure **22** illustrates a user main display in accordance with a preferred embodiment. World **2200** and local news **2210** is provided based on a user's preference. The user has also selected real estate **2230** as an item to provide direct information on the main display. Also, a different agent **2220** is provided based on the user's preference.

Figure **23** illustrates an agent interaction in accordance with a preferred embodiment. The agent **2310** is communicating information **2300** to a user indicating that the user's life insurance needs have changed and pointing the user to the chart that best summarizes the information for the user. Particular tips **2395** are provided to facilitate more detailed information based on current user statistics. A chart **2370** of the user's life insurance needs is also highlighted at the center of the display to assist the user in determining appropriate action. A button **2380** is provided to facilitate changing the policy and a set of buttons **2390** are provided to assist a user in selecting various views of the user's insurance requirements.

Event Backgrounder

An Event Backgrounder is a short description of an upcoming event that is sent to the user just before an event. The Event Backgrounder is constantly updated with the latest information related to this event. Pertinent information such as itinerary and logistics are included, and other useful information, such as people the user knows who might be in the same location, are also included. The purpose of the Event Backgrounder is to provide the most up-to-date information about an event, drawing from a number of resources, such as public web sites and the user's calendar and contact lists, to allow the user to react optimally in a given situation.

Vicinity Friend Finder

This software looks for opportunities to tell the user when a friend, family member or acquaintance is or is going to be in the same vicinity as the user. This software scans the user's calendar for upcoming events. It then uses a geographic map to

compare those calendar events with the calendar events of people who are listed in his contact list. It then informs the user of any matches, thus telling the user that someone is scheduled to be near him at a particular time.

Information Overload

The term *information overload* is now relatively understood in both its definition as well as its implications and consequences. People have a finite amount of attention that is available at any one time, but there is more and more vying for that attention every day. In short, too much information and too little time are the primary factors complicating the lives of most knowledge workers today.

The first attempts to dynamically deal with information overload were primarily focused on the intelligent filtering of information such that the *quantity* of information would be lessened. Rather than simply removing random bits of information, however, most of these approaches tried to be intelligent about what information was ultimately presented to the user. This was accomplished by evaluating each document based on the user's interests and discarding the less relevant ones. It follows, therefore, that the *quality* was also increased.

Filtering the information is only a first step in dealing with information is this new age. Arguably, just as important as the quality of the document is having ready access to it. Once you have entered a meeting, a document containing critical information about the meeting subject delivered to your office is of little value. As the speed of business continues to increase fueled by the technologies of interconnectedness, the ability to receive quality information *wherever* and *whenever* you are becomes critical. This new approach is called intelligent information delivery and is heralding in a new information age.

A preferred embodiment demonstrates the intelligent information delivery theory described above in an attempt to not only reduce information overload, but to deliver high quality information where and when users' require it. In other words, the system delivers right information to the right person at the right time and the right place.

Active Knowledge Management System Description

Figure **24** is a block diagram of an active knowledge management system in accordance with a preferred embodiment. The system consists of the following parts: back-end **2400** connection to one or more servers, personal mobile wireless clients (Awareness Machine)**2430**, **2436**, public clients (Magic Wall) **2410**, **2420**, web clients **2446**, **2448**, e-mail clients **2450**, **2460**.

Back-end Server (2400) Processes

Figure **25** is a block diagram of a back end server in accordance with a preferred embodiment. The back-end (**2400** of Figure **24**) is a computer system that has the following software active: Intelligent Agents Coordinator (Munin) **2580**, Information Prioritization Subsystem **2530**, a set of continuously and periodically running information gathering and processing Intelligent Agents **2500**, **2502** and **2504**, User Profiles Database **2542** and supporting software, Information Channels Database **2542** and supporting software, communications software **2550**, information transformation software **2560**, and auxiliary software.

The Awareness Machine (2446 & 2448 of Figure 24)

The Awareness Machine is a combination of hardware device and software application. The hardware consists of handheld personal computer and wireless communications device. The Awareness Machine reflects a constantly updated state-of-the-owner's-world by continually receiving a wireless trickle of information. This information, mined and processed by a suite of intelligent agents, consists of mail messages, news that meets each user's preferences, schedule updates, background information on upcoming meetings and events, as well as weather and traffic. The Awareness Machine is covered by another patent application.

Figure 26 is a block diagram of a magic wall in accordance with a preferred embodiment.

The Magic Wall

The Magic Wall hardware includes:

- Computer system **2640** connected to the back-end server
- Sensor array **2634**, **2630** and **2632** detects presence, position, and identity of a person
- Large touch-sensitive display **2620**
- Sound input **2610** /output **2614** hardware

The Magic Wall software supports:

- Multimedia output compatible with current Web standards
- Speech recognition
- Tactile input
- Intelligent agents representations in the form of speech-enabled animated characters

The Magic Wall operates as follows:

1. If a user appears in the vicinity of Magic Wall, the sensor array triggers "user here" event that sends an environmental cue containing the person's id and the location to the Intelligent Agent Coordinator.
2. User is identified based on the information returned by the sensor array.
3. The Magic Wall switches to "locked on the user" mode. If another user approaches, the system will notify him or her that it cannot serve another user while the current user is being served.
4. Intelligent Agent Coordinator is notified about the user presence.
5. The Intelligent Agent Coordinator decides if there is pertinent to that user and Magic Wall location time-sensitive information to show (e.g. traffic report, meeting reminder). If such information exists, it is prepared for delivery. If not, control is transferred to the Information Prioritization Subsystem.
6. Information Prioritization Subsystem decides what information is most relevant to the user based on their personal profile, freshness of the information, and the Intelligent Agent Coordinator's prior suggestions.
7. The page of information identified as the most relevant to the user at this time and place is shown. The act of the information delivery can also include animation and speech output of the intelligent agent representation.

8. If user desires so, he or she can ask Magic Wall to show a particular page. The Magic Wall recognizes the speech fragment and then identifies and shows the requested page.
9. As the user departs from the Magic Wall area, the sensor array triggers "user left" event.
10. The Magic Wall switches back to the waiting state.

Other Clients

The Web client is a standard browser navigating to a set of Web pages which allow user to see the same information that is available via the Magic Wall.

The e-mail client is any standard e-mail program.

Intelligent Agent Coordinator Description

This piece of code is the coordinating agent (or meta-agent) for the Active Knowledge Management system. This means that all communications between the system and each user, as well as communication between the different minion agents are handled (coordinated) by the Intelligent Agent Coordinator. Examples of these minion agents are:

- *BackgroundFinder* - an agent that parses meeting text determining important keywords and phrases and finds background information on the meeting for each user
- *TrafficFinder* - an agent that finds traffic information for each user based on where they live
- Several other agents that are responsible for doing statistical analysis of the data in each user's profile and updating fields pertinent to that data

The Intelligent Agent Coordinator **2580** of Figure **25** is also the user's "interface" to the system, in that whenever the user interacts with the system, regardless of the GUI or other end-user interface, they are ultimately dealing with (asking questions of or sending commands to) the Intelligent Agent Coordinator. The Intelligent Agent Coordinator has four primary responsibilities: 1) monitoring user activities, 2) handling information requests, 3) maintaining each user's profile, and 4) routing information to and from users and to and from the other respective agents.

Monitoring User Activities

Anytime a user triggers a sensor the Intelligent Agent Coordinator receives an "environmental cue." These cues not only enable the Intelligent Agent Coordinator to gain an understanding where users' are for information delivery purposes, but also to learn the standard patterns (arrival time, departure time, etc.) of each persons' life. These patterns are constantly being updated and refined in an attempt to increase the system's intelligence when delivering information. For instance, today it is not uncommon for a person to have several email accounts (work-based, home-based, mobile-based, etc.) as well as several different computers involved in the retrieval process for all of these accounts. Thus, for the Intelligent Agent Coordinator to be successful in delivering information to the correct location it must take into account all of these accounts and the times that the user is likely to be accessing them in order to maximize the probability that the user will see the information. This will be discussed further in another section.

Handling Information Requests

The Intelligent Agent Coordinator handles information requests from other agents in order to personalize information intended for each user and to more accurately reflect each user's interests in the information they are given. These requests will commonly be related to the user's profile. For instance, if an agent was preparing a traffic report for a user it may request the traffic region (search string) of that user from the Intelligent Agent Coordinator. All access to the user's profile data is accessed in this method.

Maintaining User Profiles

User profiles contain extensive information about the users. This information is a blend of user-specified data and information that the Intelligent Agent Coordinator has learned and extrapolated from each user's information and activities. In order to protect the data contained in the profiles, the Intelligent Agent Coordinator must handle all user information requests. The Intelligent Agent Coordinator is constantly modifying and updating these profiles by watching the user's activities and attempting to learn the patterns of their lives in order to assist in the more routine, mundane tasks. The Intelligent Agent Coordinator also employs other agents to glean meaning from each user's daily activities. These agents mine this data trying to discover indications of current interests, long-term interests, as well as time delivery preferences for each type of information. Another important aspect of the Intelligent Agent Coordinator's observations is that it also tries to determine where each user is physically located throughout the day for routing purposes.

Information Routing

Most people are mobile throughout their day. The Intelligent Agent Coordinator tries to be sensitive to this fact by attempting to determine, both by observation (unsupervised learning) and from cues from the environment, where users are or are likely to be located. This is certainly important for determining where to send the user's information, but also for determining in which *format* to send the information. For instance, if a user were at her desk and using the web client, the Intelligent Agent Coordinator would be receiving indications of activity from her PC and would know to send any necessary information there. In addition, because desktop PCs are generally quite powerful, a full-featured, graphically intense version could be sent. However, consider an alternative situation: the Intelligent Agent Coordinator has received an indication (via the keycard reader next to the exit) that you have just left the building. Minutes later the Intelligent Agent Coordinator also receives notification that you have received an urgent message. The Intelligent Agent Coordinator, knowing that you have left the building and having not received any other indications, assumes that you are reachable via your handheld device (for which it also knows the capabilities) and sends the text of the urgent message there, rather than a more graphically-oriented version.

Inherent Innovations

The Active Knowledge Management system represents some of the most advanced thinking in the world of knowledge management and human computer interaction. Some of the primary innovations include the following:

- The Intelligent Agent Coordinator as illustrated above.
- The development, demonstration, and realization of the theory of Intelligent Information Delivery
- Support for several channels of information delivery, all of which utilize a common back-end. For instance, if a user is in front of a Magic Wall the information will be presented in a multimedia-rich form. If the system determines that the user is mobile, the information will be sent by to their Awareness Machine in standard text. It facilitates delivery of information whenever and wherever a user requires the information.

- Personalization of information based not only on a static user profile, but also by taking into account history of the user interactions and current real-time situation including "who, where, and when" awareness.
- Utilization of fast and scalable Information Prioritization Subsystem that takes into account Intelligent Agents Coordinator opinion, user preferences, and history of user interactions. It takes the load of mundane decisions off the Intelligent Agents part therefore allowing the agents to be much more sophisticated and precise without compromising the system scalability.
- Speech recognition and speech synthesis in combination with intelligent agent animated representation and tactile input provides for efficient, intuitive, and emotionally rewarding interaction with the system.

Supporting Code in Accordance With A Preferred Embodiment

The following code is written and executed in the Microsoft Active Server Pages environment in accordance with a preferred embodiment. It consists primarily of Microsoft Jscript with some database calls embedded in the code to query and store information in the database.

Intention-Centric Interface

Create an Intention ASP Page ("intention_create.asp")

```
<%@ LANGUAGE = "JScript" %>
<%
Response.Buffer = true;
Response.Expires = 0;
%>

<html>
<head>
        <title>Create An Intention</title>
</head>

<body bgcolor="#FFE9D5" style="font-family: Arial" text="#000000">


<%
//Define some variables

upl = Server.CreateObject("SoftArtisans.FileUp")
intention_name = upl.Form("intention_name")
intention_desc = upl.Form("intention_desc")

//intention_name = Request.Form("intention_name")
//intention_desc = Request.Form("intention_desc")

//intention_icon = Request.Form("intention_icon")
```

```
submitted = upl.Form("submitted")
items = new Enumerator(upl.Form)
%>

<%
//Establish connection to the database
objConnection = Server.CreateObject("ADODB.Connection")
objConnection.Open("Maelstrom")
%>

<%
//Check to see if the person hit the button and do the appropriate thing
if (submitted == "Add/Delete")
{
        flag = "false"

        //loop through all the inputs
        while(!items.atEnd())
        {
                i = items.item()

                //if items are checked then delete them
                if(upl.Form(i) == "on")
                {
                        objConnection.Execute("delete from user_intention where intention_id =" + i);
                        objConnection.Execute("delete from intentions where intention_id =" + i);
                        objConnection.Execute("delete from tools_to_intention where intention_id =" + i)
                        flag = "true"
                }
                items.moveNext()
        }

        // if items were not deleted then insert whatever is in the text field in the database
        if(flag == "false")
        {
                intention_name_short = intention_name.replace(/ /gi,"")
                objConnection.Execute("INSERT INTO intentions (intention_name,intention_desc,intention_icon) values('" +
intention_name + "'," + intention_desc + ",'" + intention_name_short + ".gif" + "')")
                Response.write("the intention short name is " + intention_name_short);
```

```
                upl.SaveAs("E:development/asp_examples/"+ intention_name_short +".gif")
            }
        }
                // Query the database to show the most recent items.
                rsCustomersList = objConnection.Execute("SELECT * FROM intentions")
    %>
    <input type="Submit" name="return_to_mcp" value="Go to Main Control Panel" onclick="location.href='default.asp'">

    <form method="post" action="intention_create.asp" enctype="multipart/form-data" >
    <TABLE border=0>
    <tr><td colspan="2"><font face="Arial" size="+1"><b>Enter in a new intention</b></font></td></tr>

    <tr><td><font face="Arial">Name:</font></td> <td><INPUT TYPE="text" name="intention_name"></td></tr>
    <tr><td><font face="Arial">Description:</font></td><td><TEXTAREA name="intention_desc"></TEXTAREA></td></tr>
    <tr><td><font face="Arial">Icon Image:</font></td><td><INPUT TYPE="file" NAME="intention_icon" size=40></td></tr>
    <tr><td colspan="2"><INPUT type="submit" name="submitted" value="Add/Delete"></td></tr>
    </TABLE>
    <HR>
    <font face="Arial" size="+1"><b>Current Intentions</b></font>
    <TABLE>
        <tr bgcolor=E69780 align="center">
        <td>
            <FONT color="white">Delete</FONT>
        </td>
        <TD>
            <FONT color="white">Itention</FONT>
        </TD>
        <TD>
            <FONT color="white">Description</FONT>
        </TD>
        <TD>
            <FONT color="white">Image</FONT>
        </TD>
        </tr>

    <%
    // Loop over the intentions in the list
    counter = 0;
    while (!rsCustomersList.EOF)
```

```
{
%>
        <tr bgcolor="white" style="font-size: smaller">
                <td align=center>
                        <INPUT type="checkbox" name="<%=rsCustomersList("intention_id")%>">
                </TD>
                <td>
                        <%= rsCustomersList("intention_name")%>
                </td>
                <td>
                        <%= rsCustomersList("intention_desc")%>
                </td>
                <td>
                        <img src="../images/<%= rsCustomersList("intention_icon")%>">
                </td>
        </tr>

<%
counter++
rsCustomersList.MoveNext()}
%>
</TABLE>
<hr>
Available Tools
</form>
</BODY>
</HTML>
```

Retrieve Intentions List ASP Page ("intentions_list.asp")

```
<!-- #include file="include/check_authentication.inc" -->

<HTML>
<HEAD>
        <TITLE>mySite! Intentions List</TITLE>

<SCRIPT LANGUAGE="JavaScript">
        function intentionsList () {

                this.internalArray = new Array();
```

```
<%
// establish connection to the database
objConnection = Server.CreateObject("ADODB.Connection");
objConnection.Open("Maelstrom");

// create query
intentionsQuery = objConnection.Execute("SELECT * FROM intentions ORDER BY intention_name asc");
%>
// write out the options
<%
numOptions = 0
while (!intentionsQuery.EOF) {
        intentionName = intentionsQuery("intention_name");
        intentionIcon = intentionsQuery("intention_icon");
%>

        this.internalArray[<%= numOptions%>] = new Array(2);
        this.internalArray[<%= numOptions%>][0] = "<%= intentionName %>";
        this.internalArray[<%= numOptions%>][1] = "images/<%= intentionIcon %>";
<%              numOptions++; intentionsQuery.moveNext();       %>

<%      }       %>
}
numIntentions = <%= numOptions%>;
intentionArray = new intentionsList().internalArray;
function selectIntention () {
    for (i=0;i<numIntentions;i++) {
        if (IntentionsListSelect.options[i].selected) {
            intentionNameTextField.value = intentionArray[i][0];
            //intentionPicture.src = intentionArray[i][1];
            break;
        }
    }
}
</SCRIPT>

</HEAD>
```

```
<BODY BGCOLOR="<%=Session("main_background")%>" style="font-family: Arial">

<CENTER>
<!— <FORM NAME="intention_list"> —>
<TABLE FRAME="BOX" border=0 CELLPADDING="2" CELLSPACING="2">

<TR><TD COLSPAN="3" STYLE="font: 20pt arial" ALIGN="CENTER"><B>Add a mySite! Intention</B></TD></TR>

<TR><TD COLSPAN="3"> </TD></TR>

<TR>
        <TD width="100"><font size="-1">Please Select An Intention You Would Like to Add to Your List</font></TD>
        <TD colspan=2>
                <SELECT ID="IntentionsListSelect" NAME="IntentionsListSelect" SIZE="10" style="font: 9pt Arial;"
onClick="selectIntention()">
                <%
                intentionsQuery.moveFirst();
                for(j=0;j<numOptions;j++) { %>
                        <OPTION VALUE="<%= intentionsQuery("intention_id") %>" <% if (j == 0) { %> SELECTED <% }
%>>
                        <%= intentionsQuery("intention_name") %>
                        <% intentionsQuery.moveNext()
                }
                intentionsQuery.moveFirst();
                %>
                </SELECT>

        </TD>

</TR>

<TR><TD COLSPAN="3"> </TD></TR>

<TR>
        <TD width="100"><font size="-1">Customize the Intention name</font></TD>
        <TD COLSPAN=2"><INPUT TYPE="text" NAME="intentionNameTextField" ID="intentionNameTextField" SIZE="30"
VALUE="<%= intentionsQuery("intention_name") %>"></TD>
</TR>
```

```
<TR><TD COLSPAN="3"> </TD></TR>

<TR>
        <TD COLSPAN="3" ALIGN="CENTER">
                <INPUT TYPE="button" NAME="intentionOKButton" VALUE="  OK  " SIZE="10" ID="intentionOKButton"
onClick="javaScript:top.opener.top.navframe.addAnIntention();">
                              
                <INPUT TYPE="button" NAME="intentionCancelButton" VALUE="Cancel" SIZE="10"
ID="intentionCancelButton" onClick="self.close();">
        </TD>
</TR>
</TABLE>
<!-- </FORM> -->
</CENTER>

<% objConnection.Close(); %>
</BODY>
</HTML>
```

Display User Intention List ASP Page (excerpted from "navigation.asp")

```
<DIV ID="intentionlist" style="position: absolute; width:210; height:95; left: 365pt; top: -5; visibility: hidden;  font-family: Arial; font-
color: #000000; font: 8pt Arial ; " >
<DIV style="position: absolute; top:7; left:7; height:78; width:210;  z-index:2; background: <%=Session("main_background")%>;
border: solid 1pt #000000; padding: 3pt; overflow: auto; alink: black; link: black;">
<body LINK="#000000" ALINK="#000000" vlink="black">
                        <%
                        // create query

                        intentionsQuery = objConnection.Execute("SELECT user_intention.* FROM user_intention,
user_intention_to_persona WHERE user_intention_to_persona.user_persona_id = " + Session("currentUserPersona") + " AND
user_intention_to_persona.user_intention_id = user_intention.user_intention_id" );
                        numintentions = 0;
                        Response.Write("<SCRIPT>numintentions=" + intentionsQuery.RecordCount +
"</SCRIPT><TABLE cellpadding='0' width='100%' cellspacing='0'>");

                        while (!intentionsQuery.EOF)
                        {
                        %>
```

```
<TR><TD><a href="javascript:changeIntention('<%= intentionsQuery("user_intention_id")
%>','<%=numintentions%>')" onmouseover="mouseOverTab()" onmouseout="mouseOutOfTab()"><font color="Black"
face="arial" size="-2"><%= intentionsQuery("intention_custom_name") %></font></a></TD><TD><IMG align="right"
SRC="images/delete.gif" alt="Delete this intention" onClick="confirmDelete(<%= intentionsQuery("user_intention_id")
%>)"></TD></TR>
                    <%numintentions++; intentionsQuery.moveNext();          %>

                <%              }
                Response.Write("<SCRIPT>numintentions="+numintentions +"</SCRIPT>");
                %>
                <tr><td colspan="2"><hr></td></tr>
                <TR><td colspan="2"><a href="javascript:changeIntention('add ...',<%=numintentions%>);"
onmouseover="mouseOverTab()" onmouseout="mouseOutOfTab()"><font color="Black" face="arial" size="-2">add
...</font></a></td></TR>
                </table>
</body>
</DIV>
<DIV style="position: absolute; top:0; left:-5; width: 230; height:105; z-index:1; "
onmouseout="intentionlist.style.visibility='hidden'" onmouseout="intentionlist.style.visibility='hidden'"
onmouseover="intentionlist.style.visibility='hidden'"></DIV>
</DIV>
</DIV>
```

While various embodiments have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of a preferred embodiment should not be limited by any of the above described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

CLAIMS

What is claimed is:

1. A method for creating a user network interface, comprising the steps of:
(a) obtaining user profile information;
(b) discerning user intentions regarding a plurality of applications based on the profile information; and
(c) displaying the plurality of applications formatted in accordance with the user profile information on a display.

2. A method for creating a user network interface as recited in claim 1, including the step of prompting a user for profile information and storing the information in a database.

3. A method for creating a user network interface as recited in claim 1, including the step of tracking user interactions with applications and recording behavioral statistics in the user profile.

4. A method for creating a user network interface as recited in claim 1, including the step of providing a common user interface for each of the plurality of applications tuned to a user's preferences.

5. A method for creating a user network interface as recited in claim 1, wherein the plurality of applications comprise: electronic commerce, information management and information delivery.

6. A method for creating a user network interface as recited in claim 1, including the step of building a customized template for user interactions with an application utilizing the profile information.

7. A method for creating a user network interface as recited in claim 7, including the step of further customizing the template based on the network provider information.

8. A method for creating a user network interface as recited in claim 7, including completint the template utilizing product information, advise and progress reports.

9. A method for creating a user network interface as recited in claim 1, including the step of agent facilitation of logic based on a user's profile information.

10. An apparatus that creates an information summary, comprising;
(a) a processor;
(b) a memory that stores information under the control of the processor;
(c) logic that obtains user profile information;
(d) logic that discerns user intentions regarding a plurality of applications based on the profile information; and
(e) logic that displays the plurality of applications formatted in accordance with the user profile information on a display.

11. A computer program embodied on a computer-readable medium that creates an information summary, comprising:

(a) a code segment that obtains user profile information;

(b) a code segment that discerns user intentions regarding a plurality of applications based on the profile information; and

(c) a code segment that displays the plurality of applications formatted in accordance with the user profile information on a display.

12. A computer program embodied on a computer-readable medium that creates an information summary as recited in claim 11, including logic that prompts a user for profile information and storing the information in a database.

13. A computer program embodied on a computer-readable medium that creates an information summary as recited in claim 11, including logic that tracks user interactions with applications and recording behavioral statistics in the user profile.

14. A computer program embodied on a computer-readable medium that creates an information summary as recited in claim 11, including logic that provides a common user interface for each of the plurality of applications tuned to a user's preferences.

15. A computer program embodied on a computer-readable medium that creates an information summary as recited in claim 11, wherein the plurality of applications comprise: electronic commerce, information management and information delivery.

16. A computer program embodied on a computer-readable medium that creates an information summary as recited in claim 11, including logic that builds a customized template for user interactions with an application utilizing the profile information.

17. A computer program embodied on a computer-readable medium that creates an information summary as recited in claim 11, including logic that further customizes the template based on the network provider information.

18. A computer program embodied on a computer-readable medium that creates an information summary as recited in claim 11, including logic that completes the template utilizing product information, advise and progress reports.

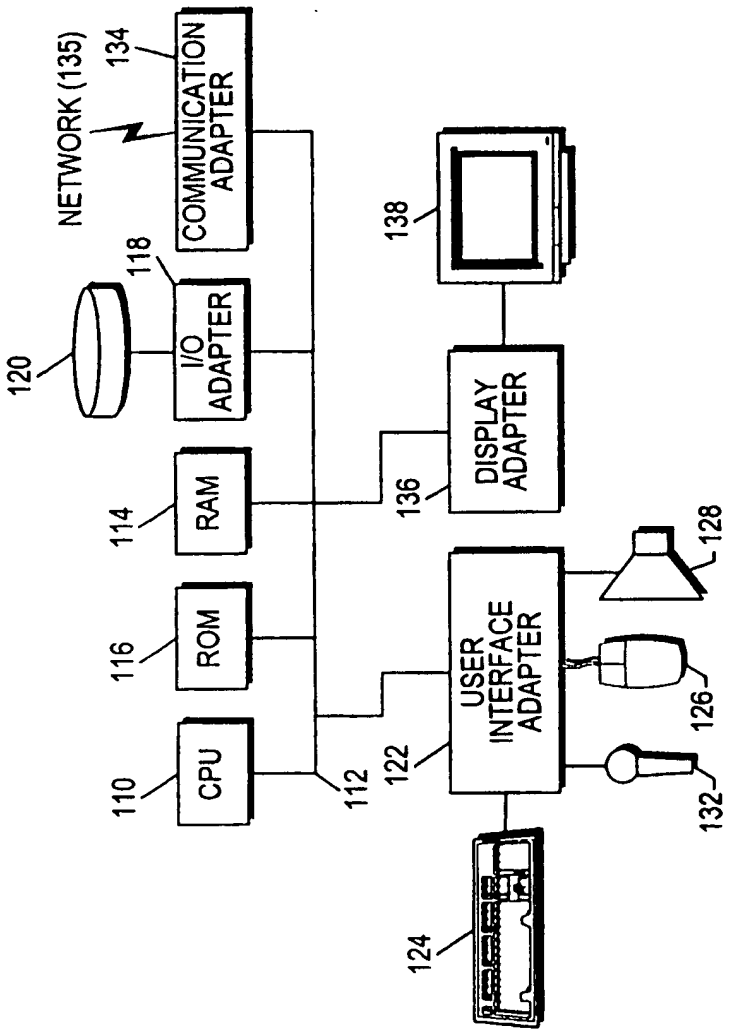

NETWORK (135)
134
COMMUNICATION ADAPTER
118
I/O ADAPTER
120
RAM
114
ROM
116
CPU
110
112
138
DISPLAY ADAPTER
136
USER INTERFACE ADAPTER
122
128
126
132
124

FIG. 1

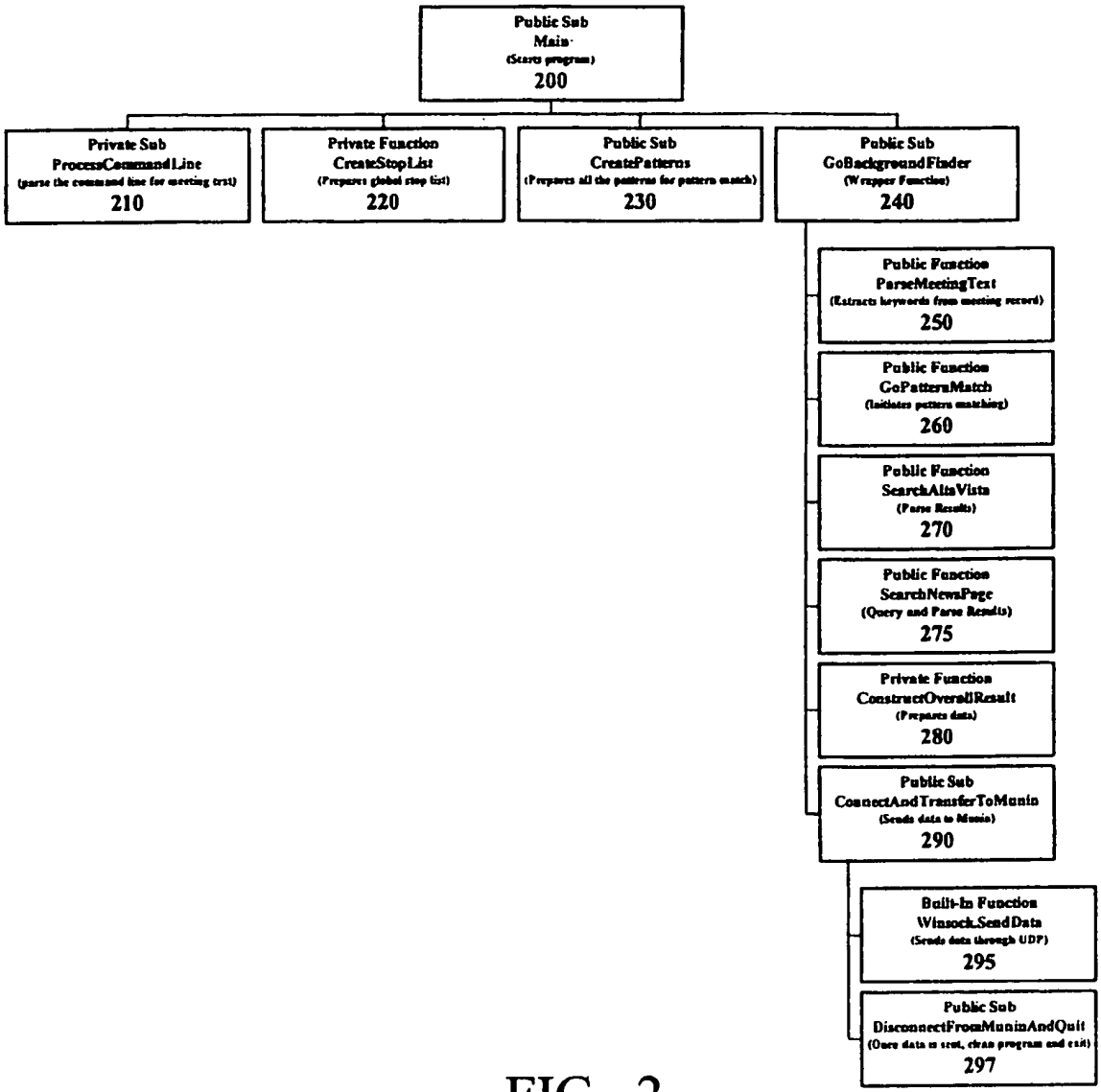

Public Sub
Main
(Starts program)
200
Private Sub
ProcessCommandLine
(parse the command line for meeting text)
210
Private Function
CreateStopList
(Prepares global stop list)
220
Public Sub
CreatePatterns
(Prepares all the patterns for pattern match)
230
Public Sub
GoBackgroundFinder
(Wrapper Function)
240
Public Function
ParseMeetingText
(Extracts keywords from meeting record)
250
Public Function
GoPatternMatch
(Initiates pattern matching)
260
Public Function
SearchAltaVista
(Parse Results)
270
Public Function
SearchNewsPage
(Query and Parse Results)
275
Private Function
ConstructOverallResult
(Prepares data)
280
Public Sub
ConnectAndTransferToMunin
(Sends data to Munin)
290
Built-In Function
Winsock.SendData
(Sends data through UDP)
295
Public Sub
DisconnectFromMuninAndQuit
(Store data in text, clean program and exit)
297

FIG. 2

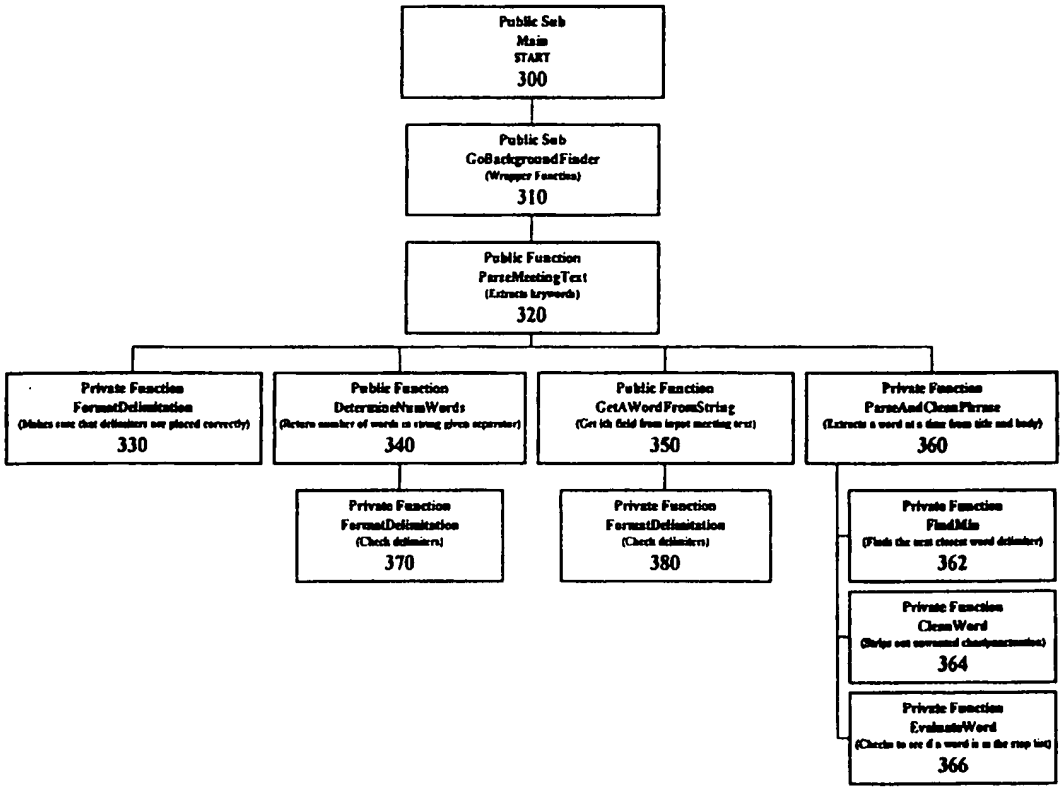

Public Sub
Main
START
300
Public Sub
GoBackgroundFinder
(Wrapper Function)
310
Public Function
ParseMeetingText
(Extracts keywords)
320
Private Function
FormatDelimitation
330
Public Function
DetermineNumWords
340
Public Function
GetAWordFromString
350
Private Function
ParseAndCleanPhrase
360
Private Function
FormatDelimitation
(Check delimiters)
370
Private Function
FormatDelimitation
(Check delimiters)
380
Private Function
FindMin
362
Private Function
CleanWord
364
Private Function
EvaluateWord
366

FIG. 3

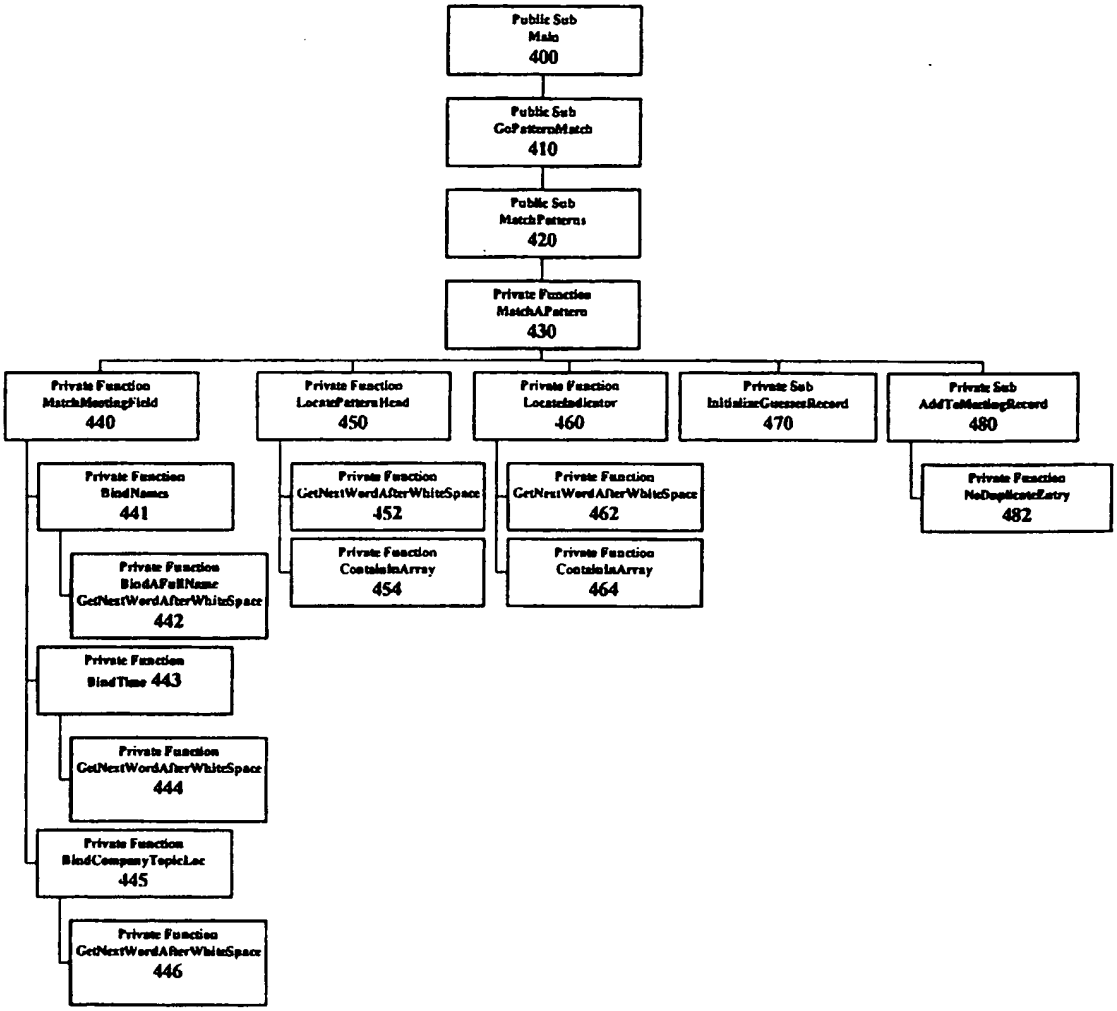

Public Sub
Main
400
Public Sub
GoPatternMatch
410
Public Sub
MatchPatterns
420
Private Function
MatchAPattern
430
Private Function
MatchMeetingField
440
Private Function
LocatePatternHead
450
Private Function
LocateIndicator
460
Private Sub
InitializeGuessesRecord
470
Private Sub
AddToMeetingRecord
480
Private Function
BindNames
441
Private Function
BindAFullName
GetNextWordAfterWhiteSpace
442
Private Function
BindTime 443
Private Function
GetNextWordAfterWhiteSpace
444
Private Function
BindCompanyTopicLoc
445
Private Function
GetNextWordAfterWhiteSpace
446
Private Function
GetNextWordAfterWhiteSpace
452
Private Function
ContainsInArray
454
Private Function
GetNextWordAfterWhiteSpace
462
Private Function
ContainsInArray
464
Private Function
NoDuplicateEntry
482

FIG. 4

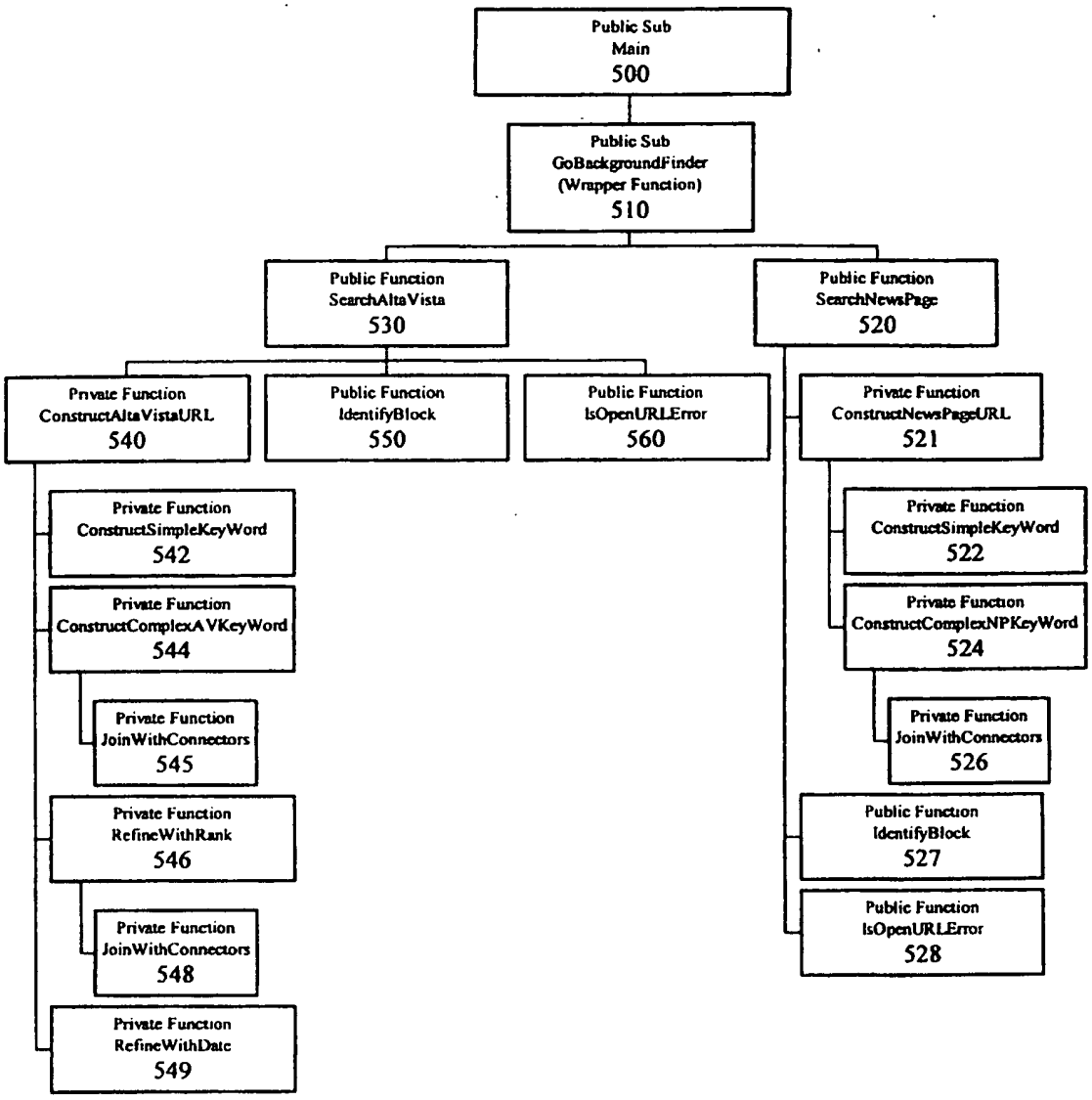

Public Sub
Main
500
Public Sub
GoBackgroundFinder
(Wrapper Function)
510
Public Function
SearchAltaVista
530
Public Function
SearchNewsPage
520
Private Function
ConstructAltaVistaURL
540
Public Function
IdentifyBlock
550
Public Function
IsOpenURLError
560
Private Function
ConstructNewsPageURL
521
Private Function
ConstructSimpleKeyWord
542
Private Function
ConstructComplexAVKeyWord
544
Private Function
JoinWithConnectors
545
Private Function
RefineWithRank
546
Private Function
JoinWithConnectors
548
Private Function
RefineWithDate
549
Private Function
ConstructSimpleKeyWord
522
Private Function
ConstructComplexNPKeyWord
524
Private Function
JoinWithConnectors
526
Public Function
IdentifyBlock
527
Public Function
IsOpenURLError
528

FIG. 5

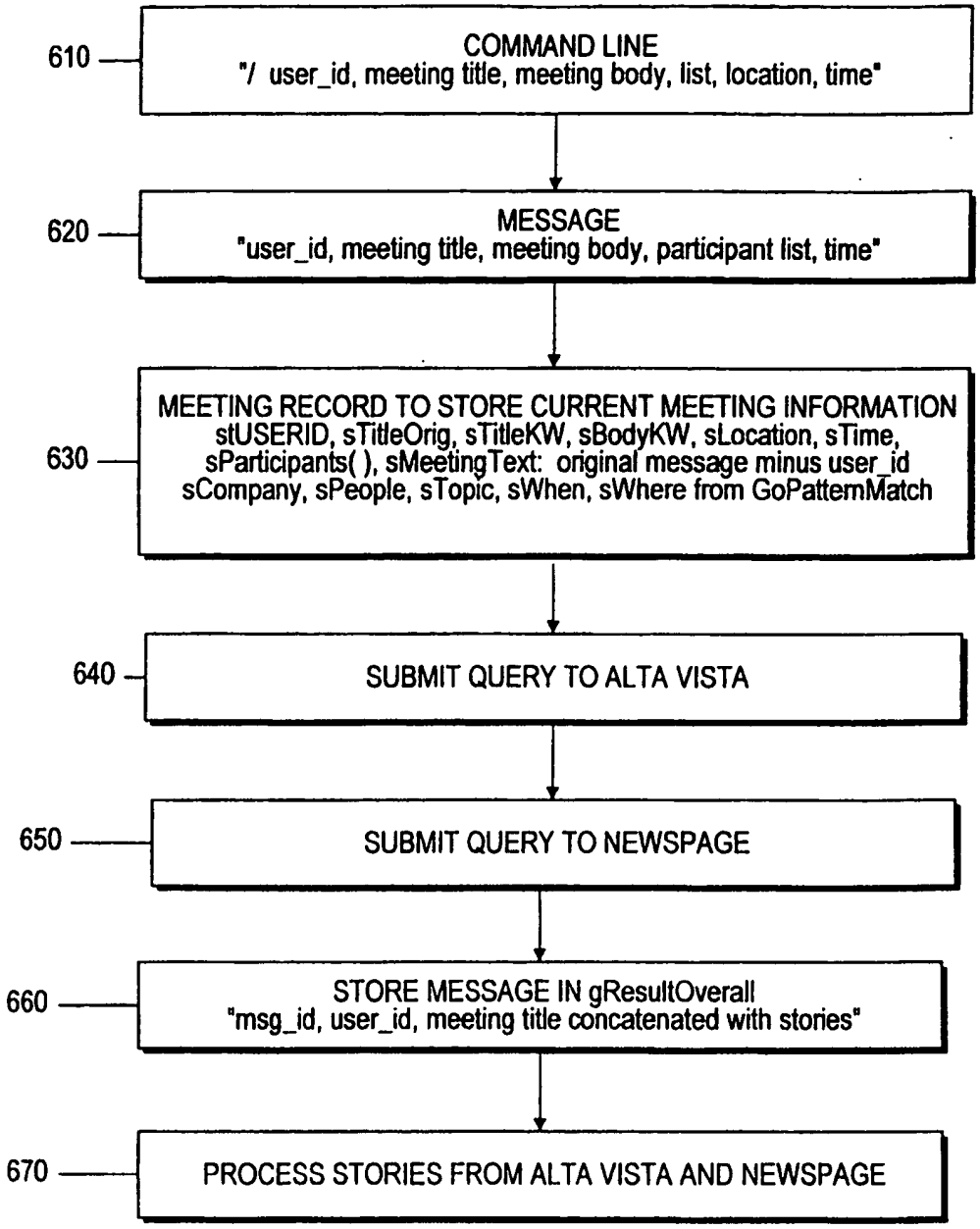

610
COMMAND LINE
"/ user_id, meeting title, meeting body, list, location, time"
620
MESSAGE
"user_id, meeting title, meeting body, participant list, time"
630
MEETING RECORD TO STORE CURRENT MEETING INFORMATION
stUSERID, sTitleOrig, sTitleKW, sBodyKW, sLocation, sTime,
sParticipants( ), sMeetingText: original message minus user_id
sCompany, sPeople, sTopic, sWhen, sWhere from GoPatternMatch
640
SUBMIT QUERY TO ALTA VISTA
650
SUBMIT QUERY TO NEWSPAGE
660
STORE MESSAGE IN gResultOverall
"msg_id, user_id, meeting title concatenated with stories"
670
PROCESS STORIES FROM ALTA VISTA AND NEWSPAGE

FIG. 6

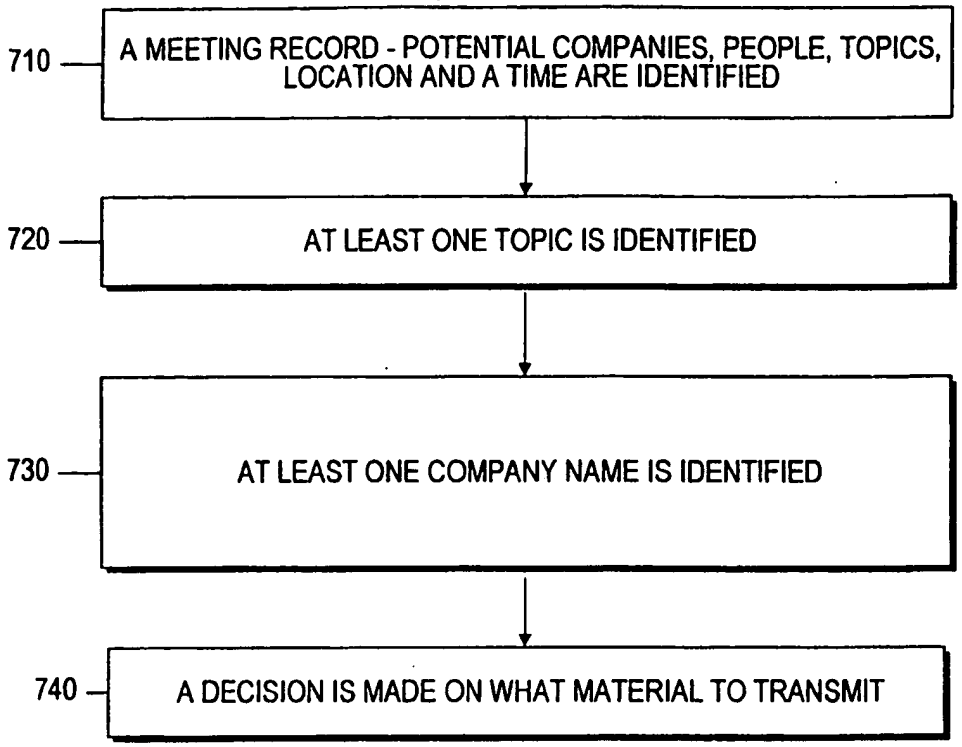

710
A MEETING RECORD - POTENTIAL COMPANIES, PEOPLE, TOPICS, LOCATION AND A TIME ARE IDENTIFIED
720
AT LEAST ONE TOPIC IS IDENTIFIED
730
AT LEAST ONE COMPANY NAME IS IDENTIFIED
740
A DECISION IS MADE ON WHAT MATERIAL TO TRANSMIT

FIG. 7

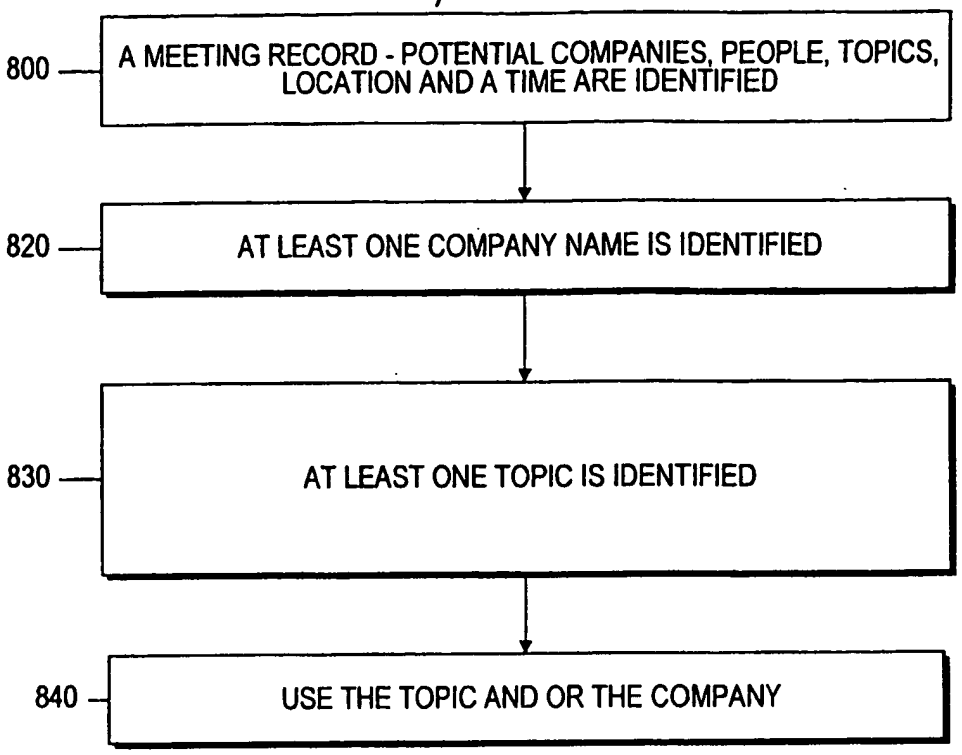

800
A MEETING RECORD - POTENTIAL COMPANIES, PEOPLE, TOPICS, LOCATION AND A TIME ARE IDENTIFIED
820
AT LEAST ONE COMPANY NAME IS IDENTIFIED
830
AT LEAST ONE TOPIC IS IDENTIFIED
840
USE THE TOPIC AND OR THE COMPANY

FIG. 8

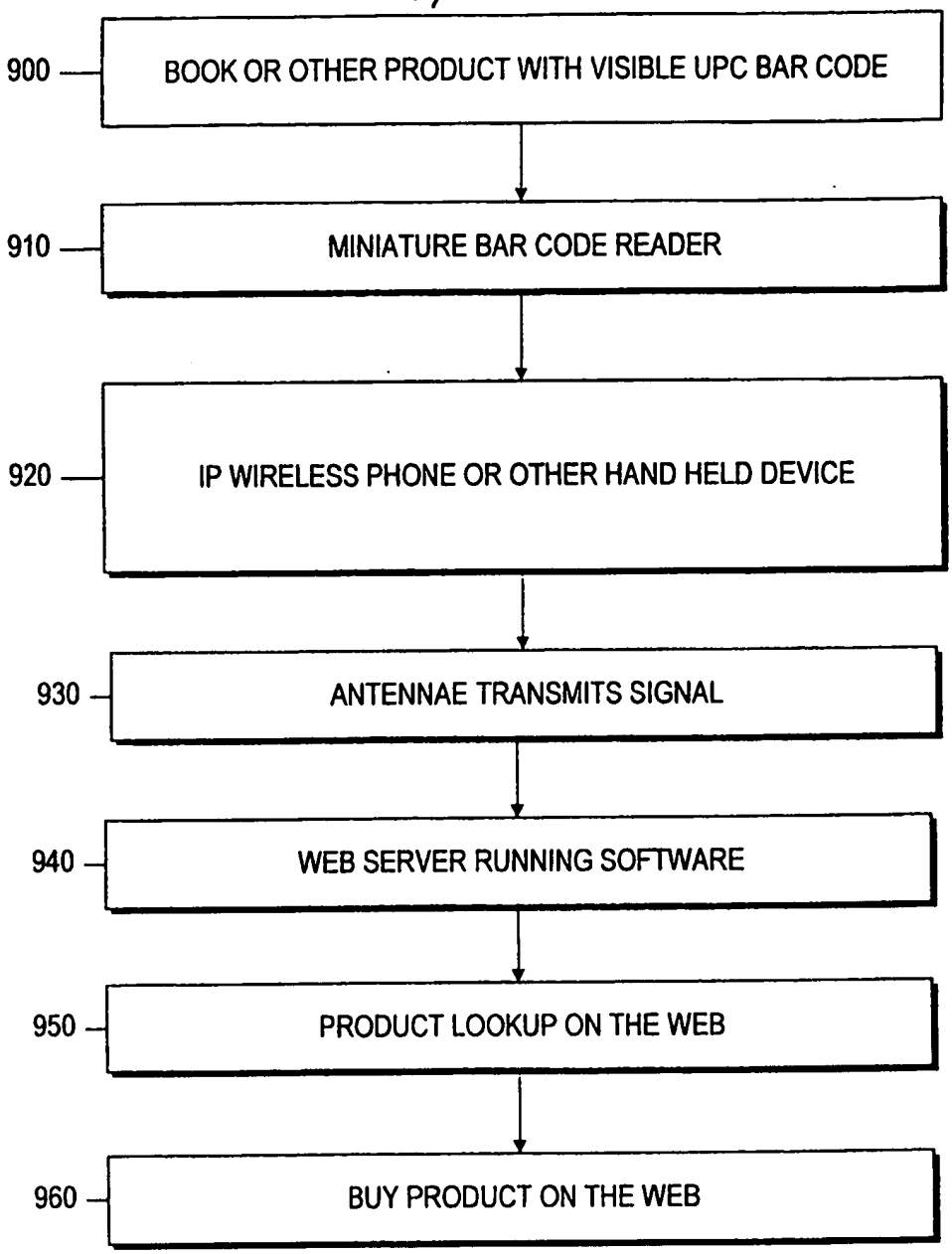

900
BOOK OR OTHER PRODUCT WITH VISIBLE UPC BAR CODE
910
MINIATURE BAR CODE READER
920
IP WIRELESS PHONE OR OTHER HAND HELD DEVICE
930
ANTENNAE TRANSMITS SIGNAL
940
WEB SERVER RUNNING SOFTWARE
950
PRODUCT LOOKUP ON THE WEB
960
BUY PRODUCT ON THE WEB

FIG. 9

FIG. 10A

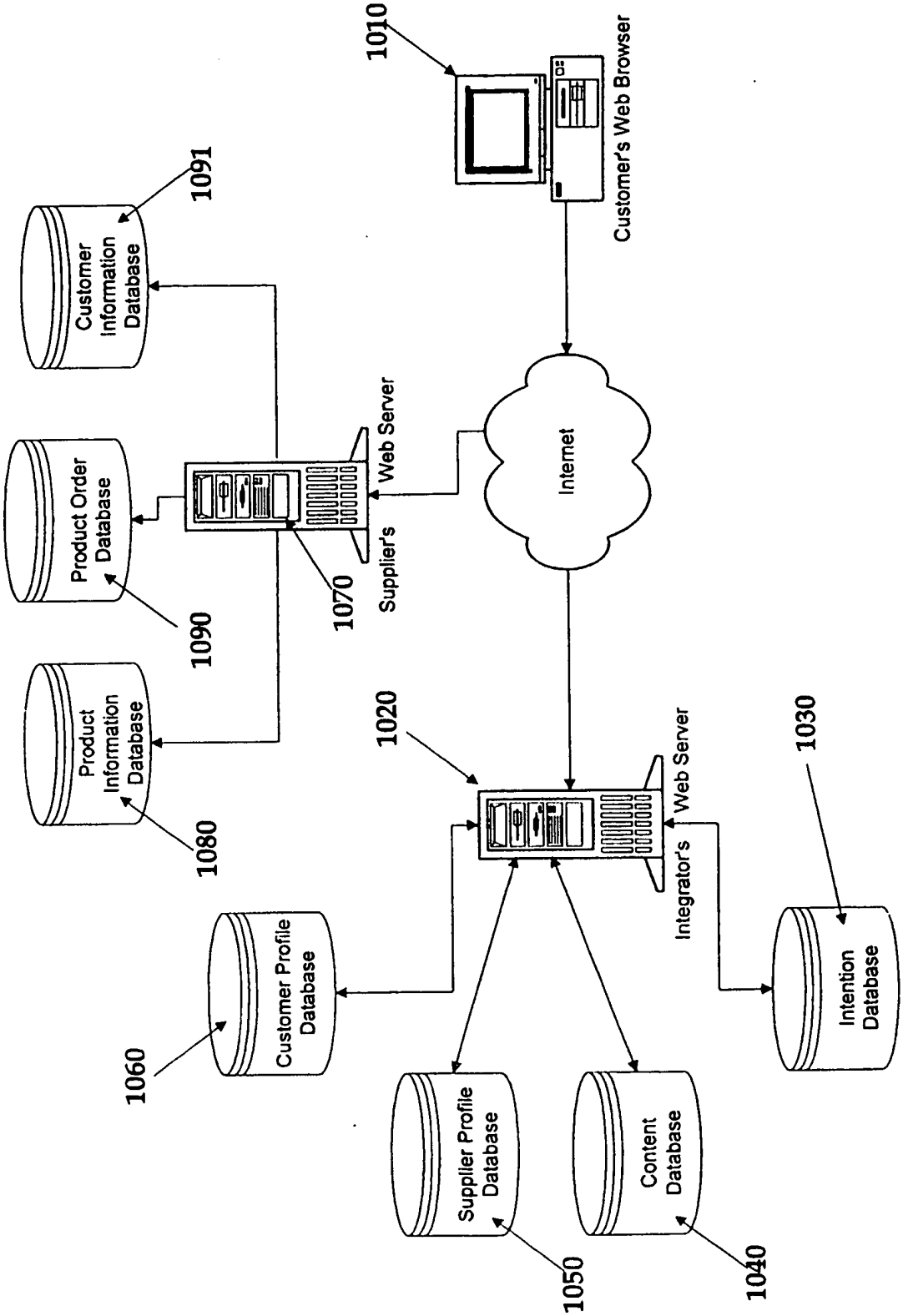

1010
Customer's Web Browser
1091
Customer Information Database
Internet
Web Server
Supplier's
1070
Product Order Database
1090
Product Information Database
1080
1020
Web Server
Integrator's
1030
Customer Profile Database
1060
Supplier Profile Database
1050
Content Database
1040
Intention Database

Fig. 10B

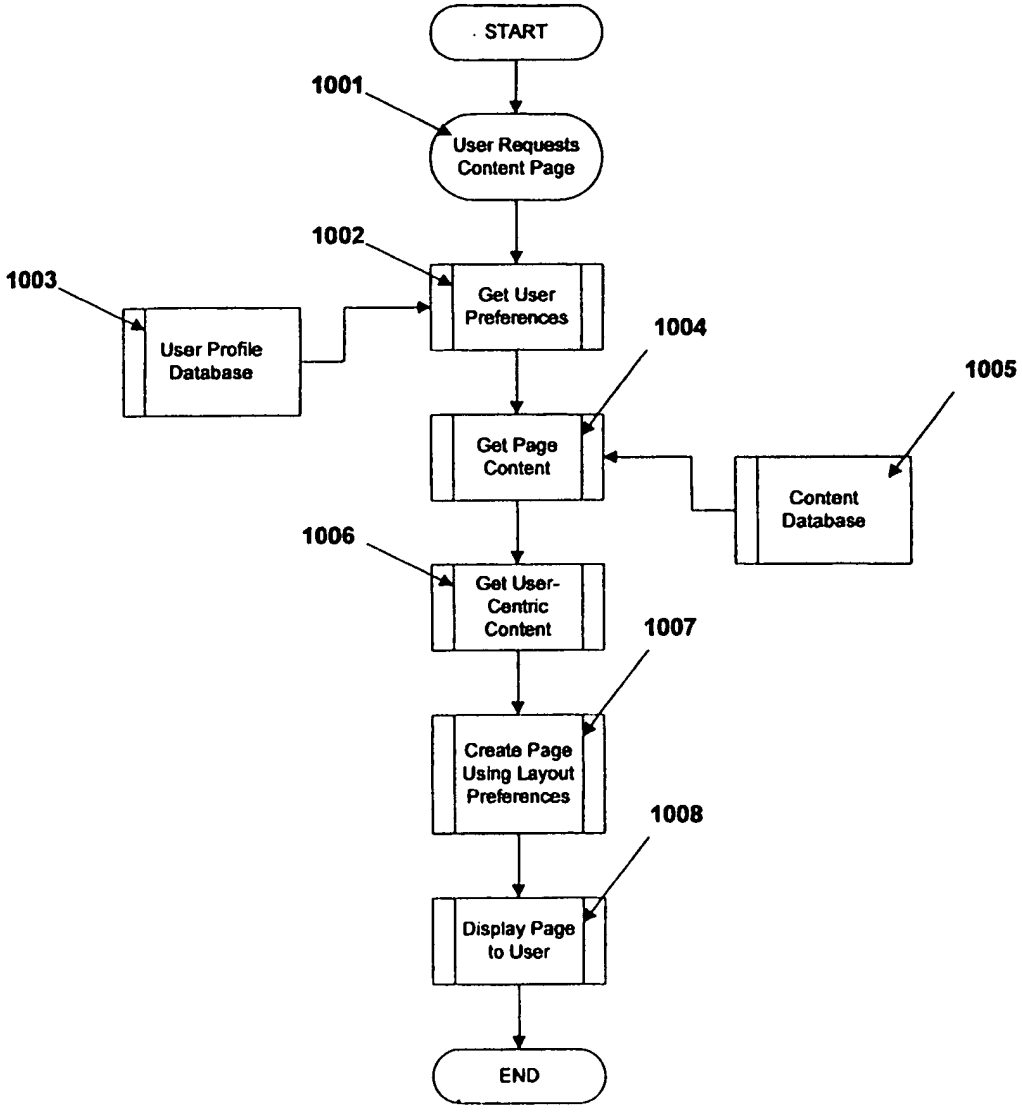

START
1001
User Requests Content Page
1002
1003
User Profile Database
Get User Preferences
1004
1005
Get Page Content
Content Database
1006
Get User-Centric Content
1007
Create Page Using Layout Preferences
1008
Display Page to User
END

Fig. 11

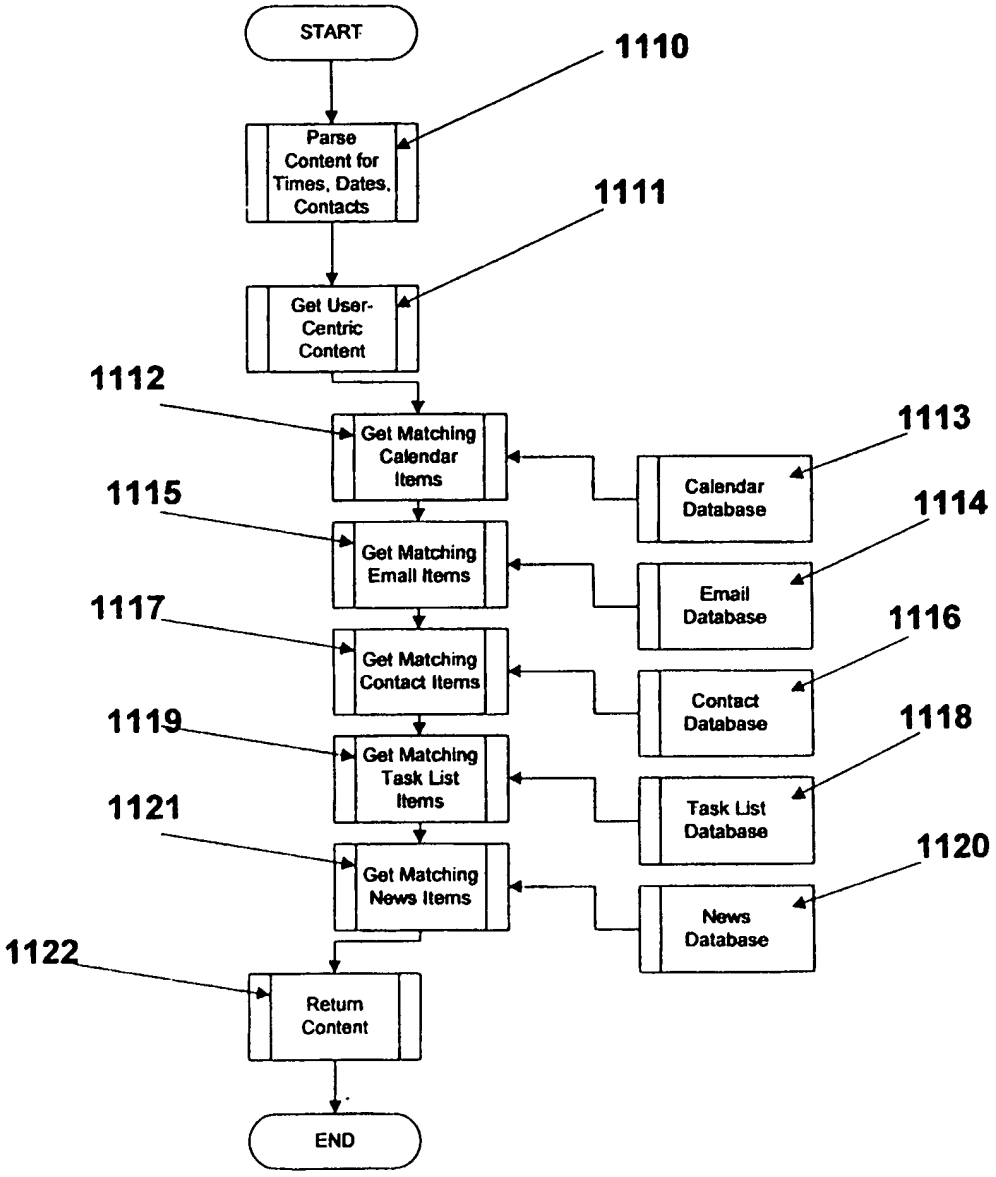

START
1110
Parse Content for Times, Dates, Contacts
1111
Get User-Centric Content
1112
Get Matching Calendar Items
1113
Calendar Database
1115
Get Matching Email Items
1114
Email Database
1117
Get Matching Contact Items
1116
Contact Database
1119
Get Matching Task List Items
1118
Task List Database
1121
Get Matching News Items
1120
News Database
1122
Return Content
END

Fig. 12

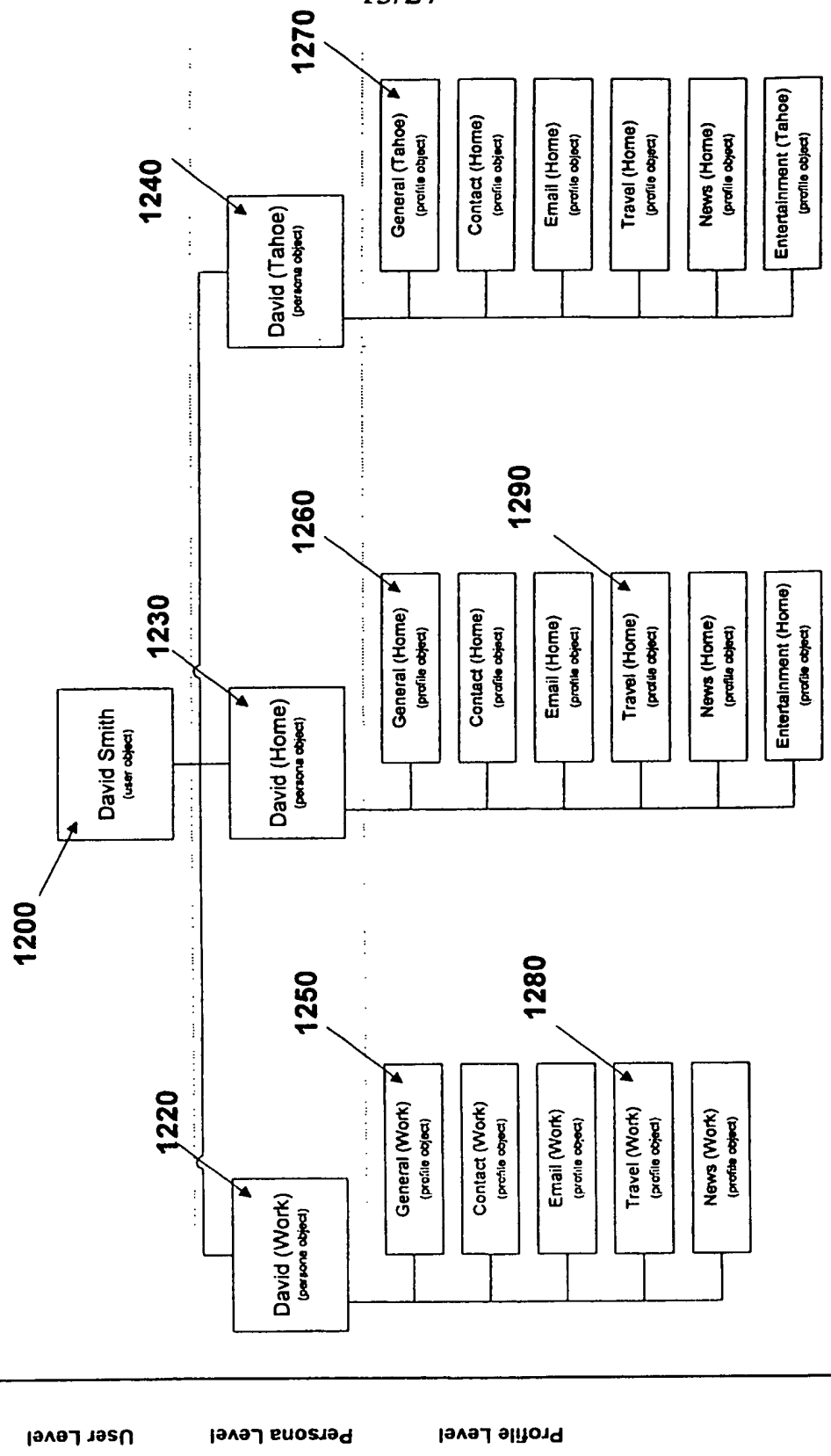

1200
David Smith
(user object)
1220
David (Work)
(persona object)
1230
David (Home)
(persona object)
1240
David (Tahoe)
(persona object)
1250
General (Work)
(profile object)
Contact (Work)
(profile object)
Email (Work)
(profile object)
Travel (Work)
(profile object)
News (Work)
(profile object)
1260
General (Home)
(profile object)
Contact (Home)
(profile object)
Email (Home)
(profile object)
1290
Travel (Home)
(profile object)
News (Home)
(profile object)
Entertainment (Home)
(profile object)
1270
General (Tahoe)
(profile object)
Contact (Home)
(profile object)
Email (Home)
(profile object)
Travel (Home)
(profile object)
News (Home)
(profile object)
Entertainment (Tahoe)
(profile object)
1280
User Level
Persona Level
Profile Level

Fig. 13

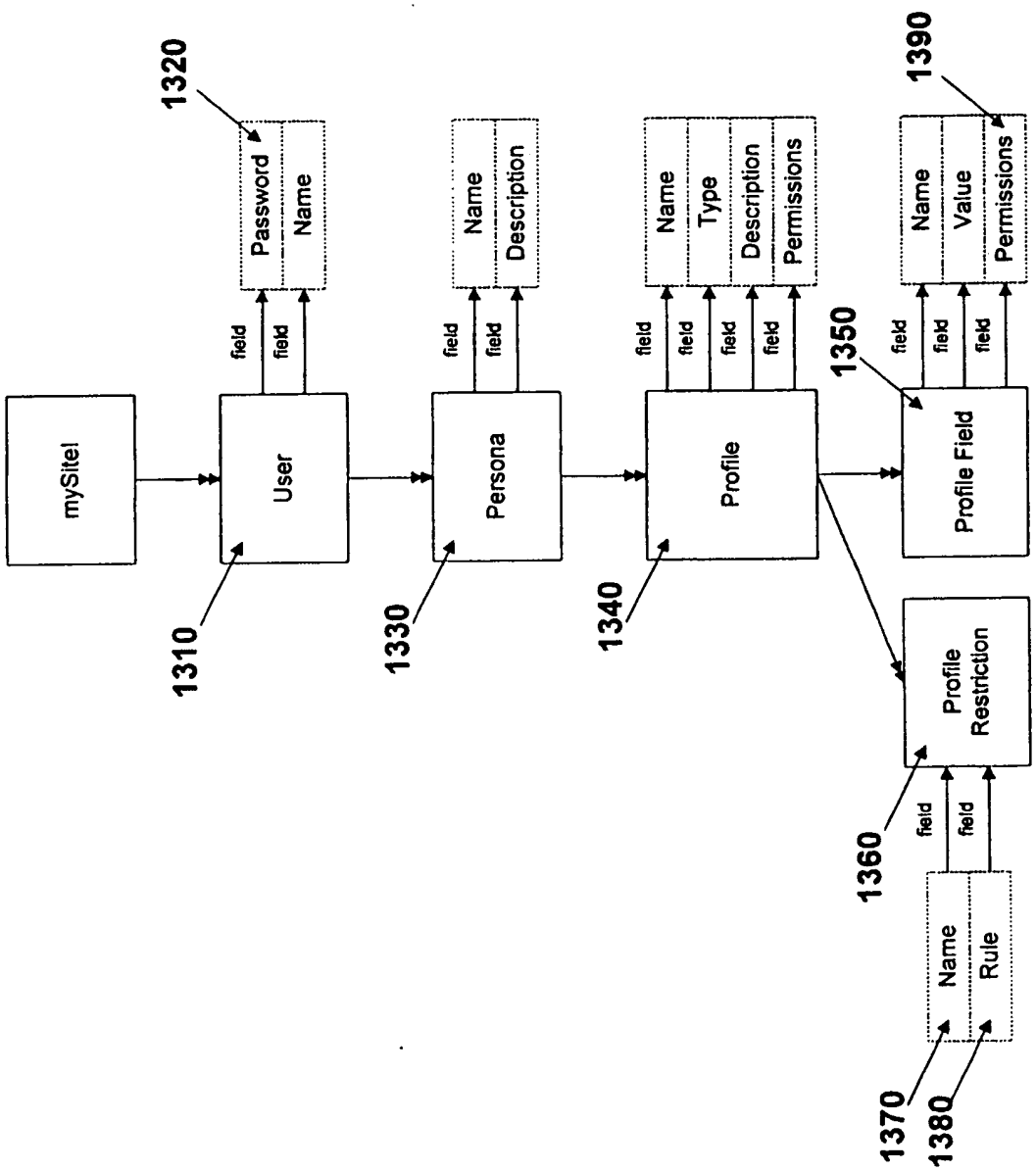

mySitel
User
Persona
Profile
Profile Field
Profile Restriction
field
Password
Name
Description
Type
Permissions
Value
Rule
1310
1320
1330
1340
1350
1360
1370
1380
1390

Fig. 14

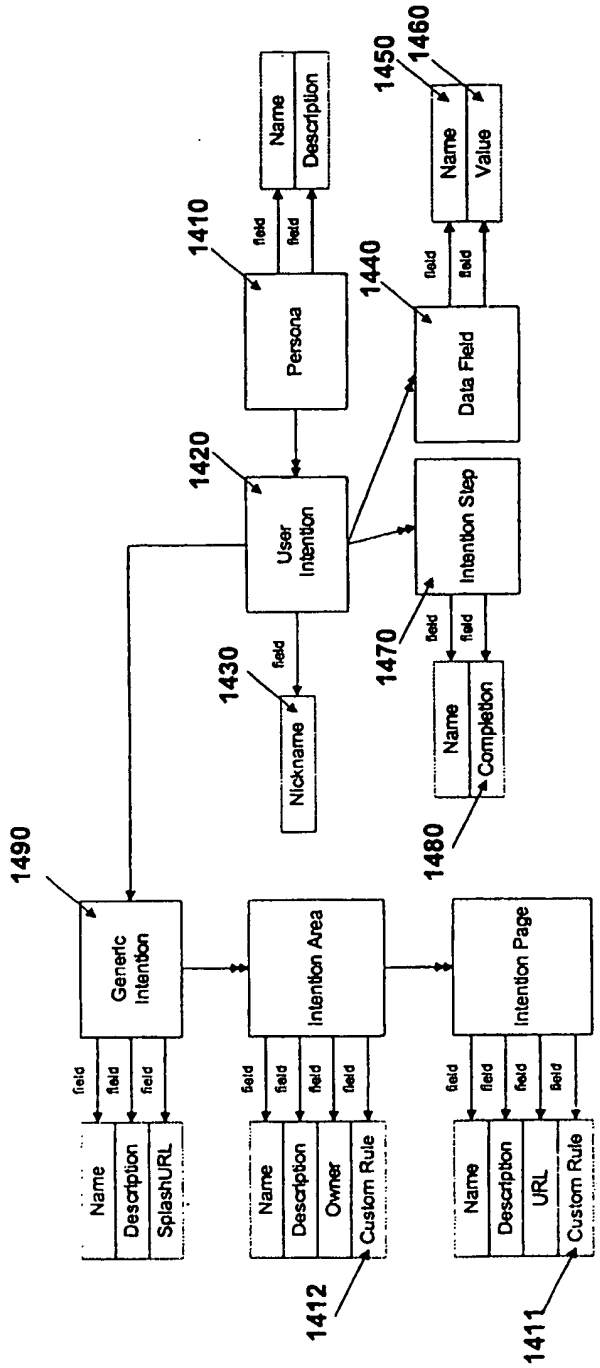

1490
Generic Intention
Name
Description
SplashURL
Intention Area
Name
Description
Owner
Custom Rule
1412
Intention Page
Name
Description
URL
Custom Rule
1411
1480
Name
Completion
1470
Intention Step
1420
User Intention
1430
Nickname
1410
Persona
Name
Description
1440
Data Field
1450
1460
Name
Value
field

Fig. 15

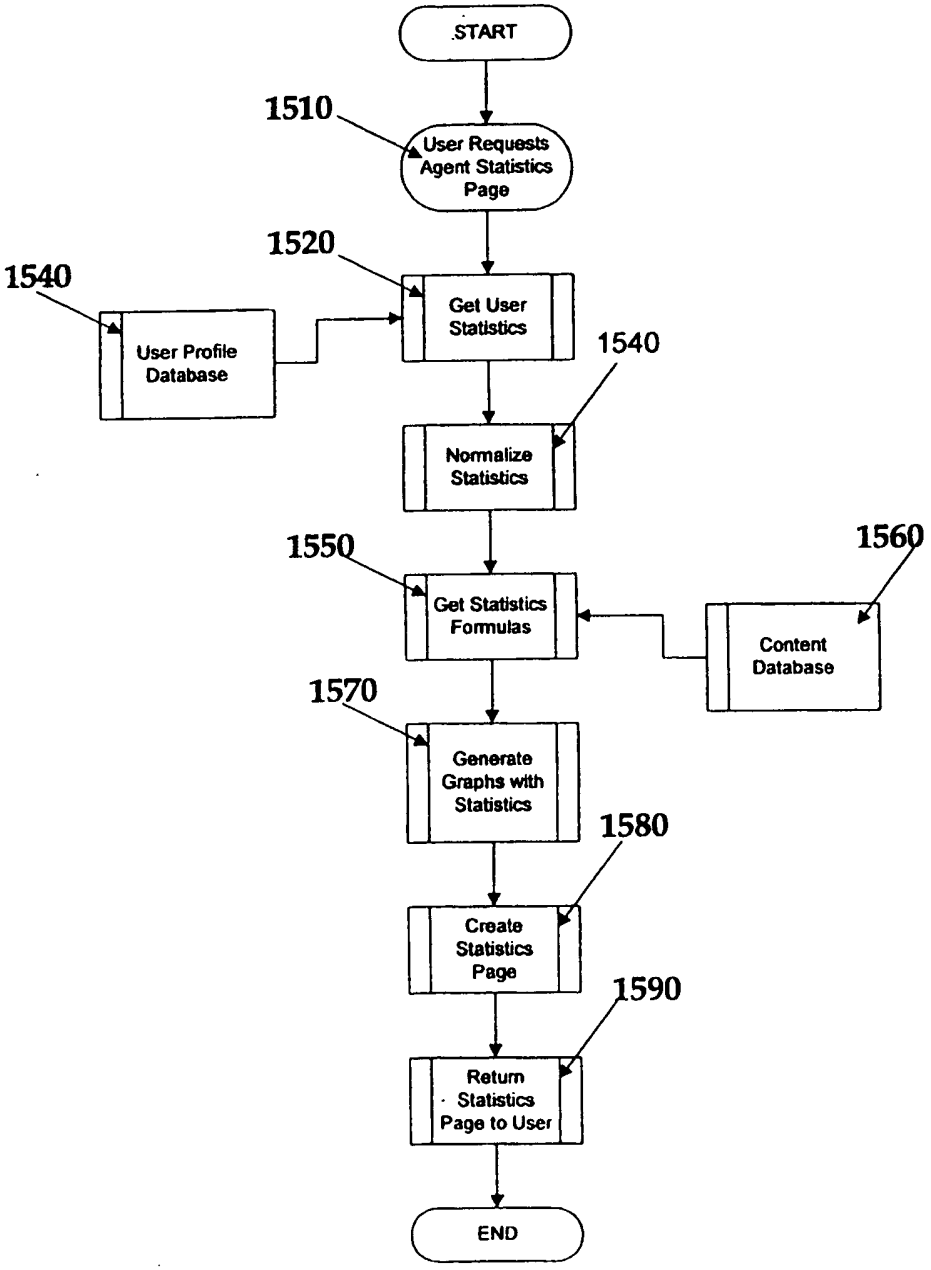

START
1510
User Requests Agent Statistics Page
1520
1540
Get User Statistics
User Profile Database
1540
Normalize Statistics
1550
1560
Get Statistics Formulas
Content Database
1570
Generate Graphs with Statistics
1580
Create Statistics Page
1590
Return Statistics Page to User
END

Fig. 16

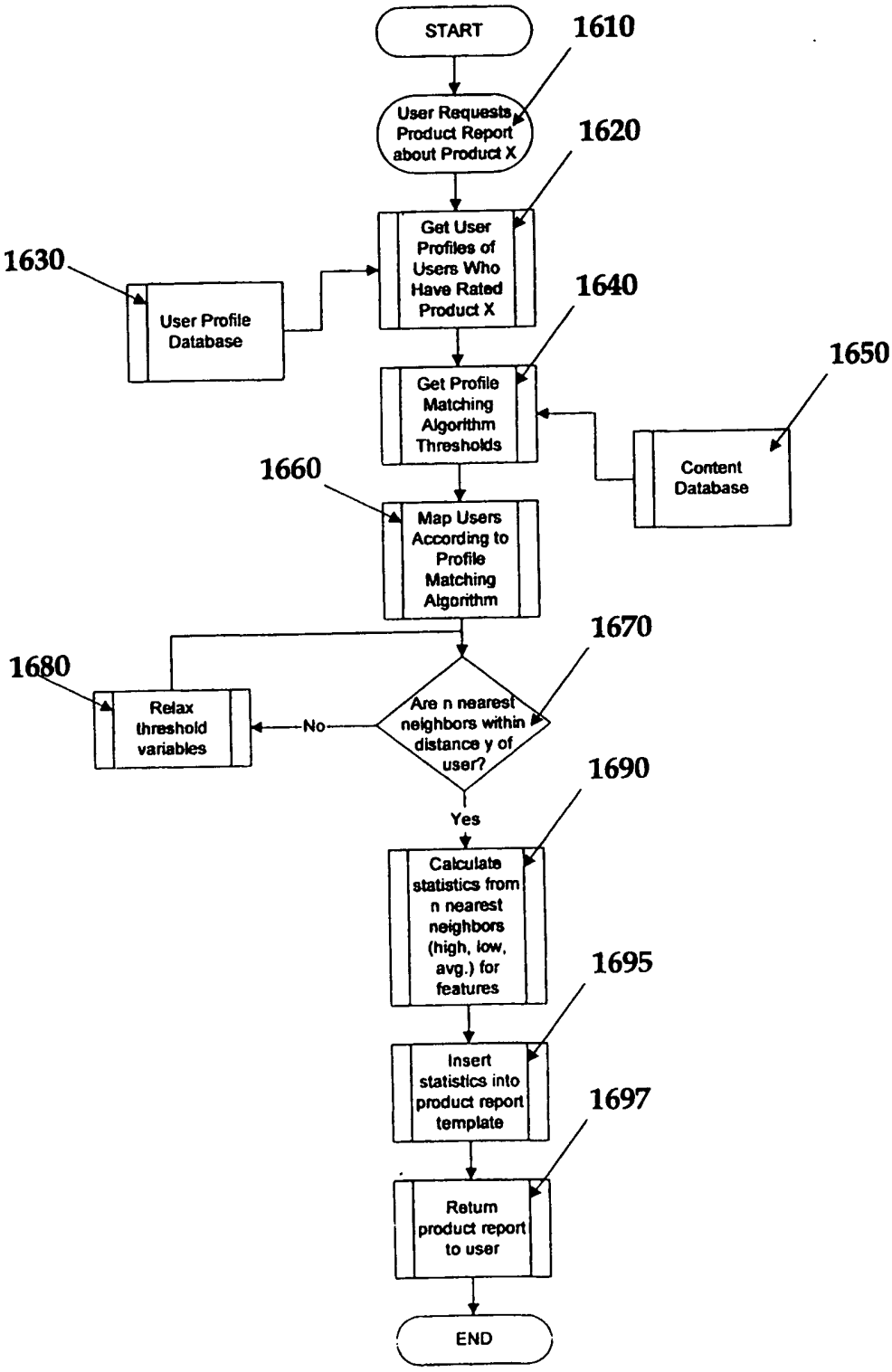

START
1610
User Requests Product Report about Product X
1620
Get User Profiles of Users Who Have Rated Product X
1630
User Profile Database
1640
Get Profile Matching Algorithm Thresholds
1650
Content Database
1660
Map Users According to Profile Matching Algorithm
1670
Are n nearest neighbors within distance y of user?
No
1680
Relax threshold variables
Yes
1690
Calculate statistics from n nearest neighbors (high, low, avg.) for features
1695
Insert statistics into product report template
1697
Return product report to user
END

Fig. 17

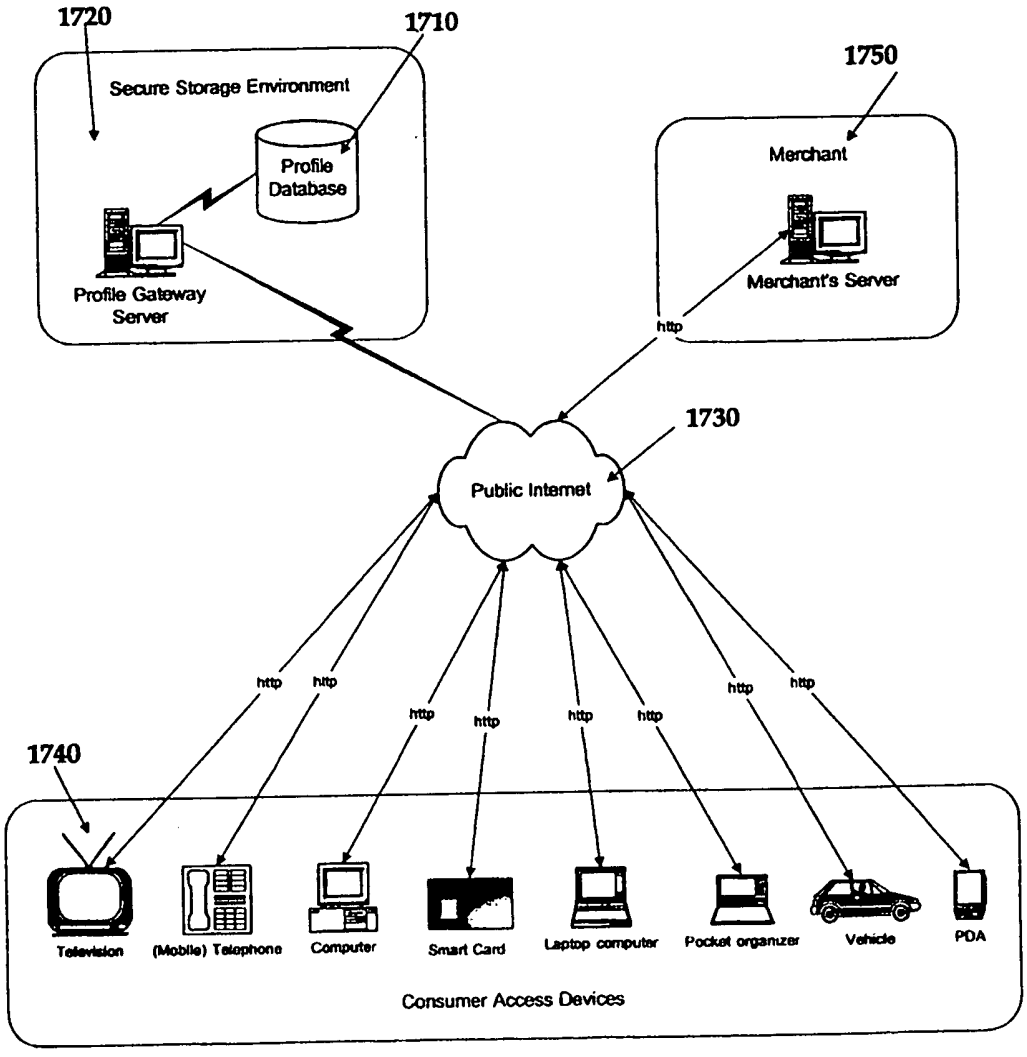

1720
1710
1750
Secure Storage Environment
Profile Database
Profile Gateway Server
Merchant
Merchant's Server
http
1730
Public Internet
http
http
http
http
http
http
http
http
1740
Television
(Mobile) Telephone
Computer
Smart Card
Laptop computer
Pocket organizer
Vehicle
PDA
Consumer Access Devices

Fig. 18

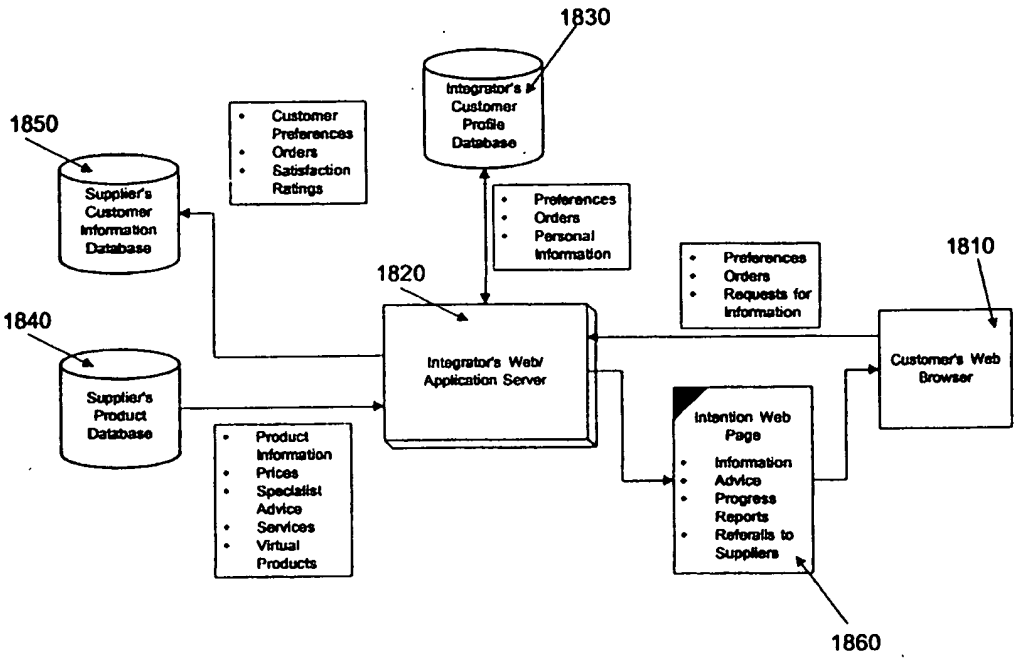
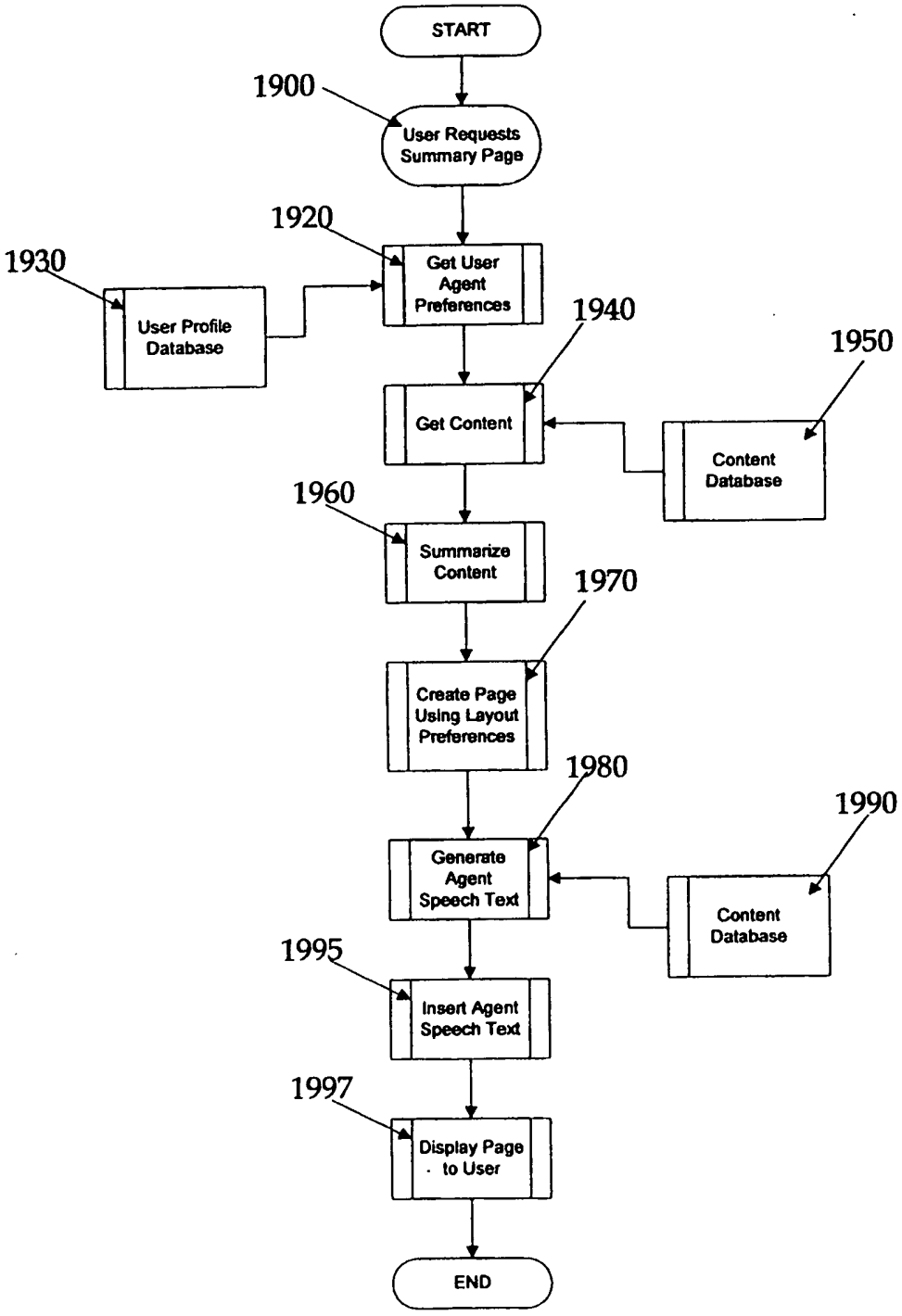

1830
Integrator's Customer Profile Database
1850
Supplier's Customer Information Database
Customer Preferences
Orders
Satisfaction Ratings
Preferences
Orders
Personal Information
1820
Preferences
Orders
Requests for Information
1810
1840
Supplier's Product Database
Integrator's Web/ Application Server
Customer's Web Browser
Product Information
Prices
Specialist Advice
Services
Virtual Products
Intention Web Page
Information
Advice
Progress Reports
Referrals to Suppliers
1860

FIG. 19


START
1900
User Requests Summary Page
1920
1930
Get User Agent Preferences
User Profile Database
1940
1950
Get Content
Content Database
1960
Summarize Content
1970
Create Page Using Layout Preferences
1980
1990
Generate Agent Speech Text
Content Database
1995
Insert Agent Speech Text
1997
Display Page to User
END

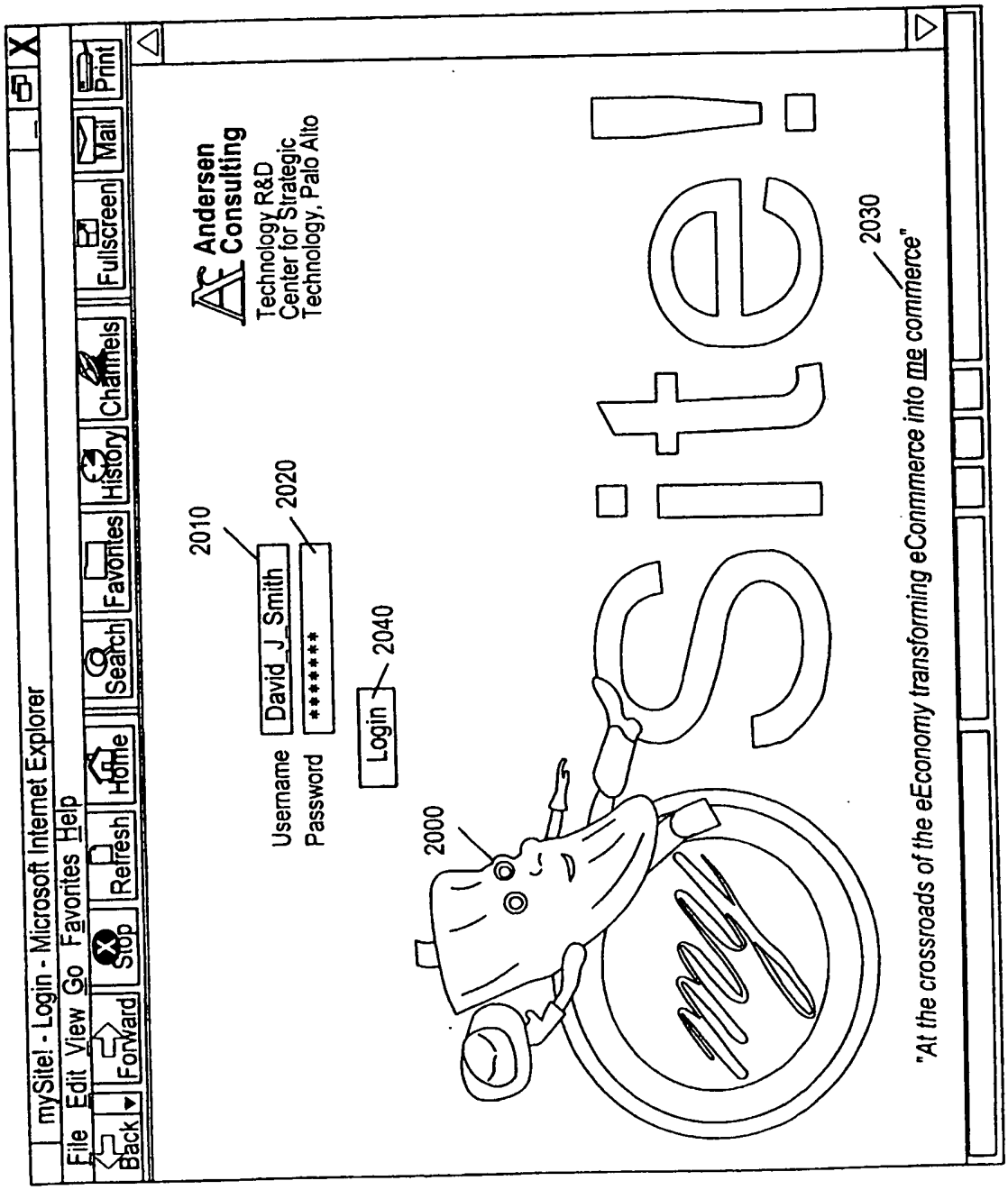

mySite! - Login - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Andersen Consulting
Technology, R&D
Center for Strategic
Technology, Palo Alto
mySite!
2010
2020
2030
2040
2000
Username
Password
David_J_Smith
****
Login
"At the crossroads of the eEconomy transforming eCommerce into me commerce"

FIG. 20

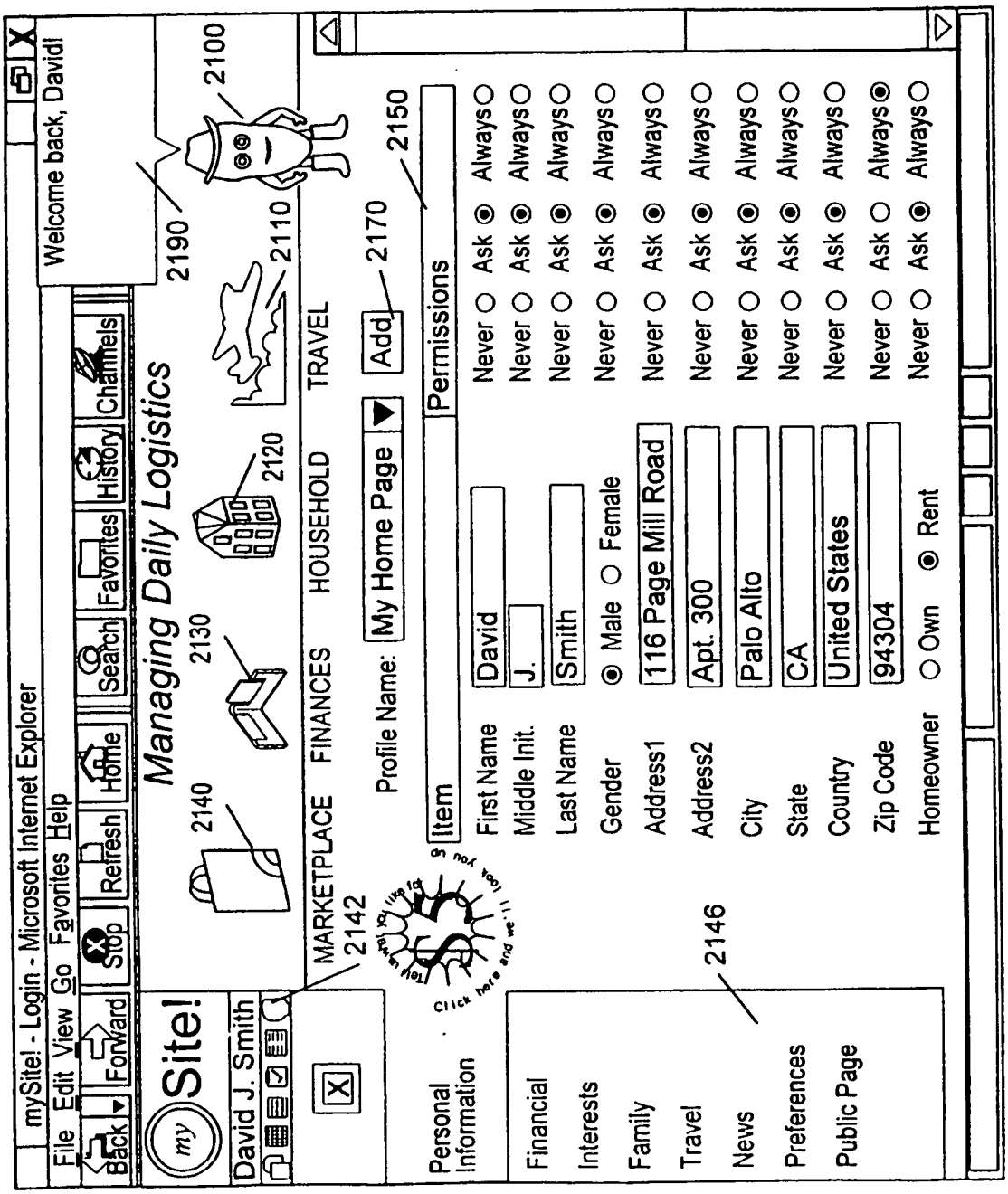

mySite! - Login - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels
my Site!
Managing Daily Logistics
Welcome back, David!
David J. Smith
MARKETPLACE FINANCES HOUSEHOLD TRAVEL
2100
2110
2120
2130
2140
2142
2146
2150
2170
2190
Personal Information
Financial
Interests
Family
Travel
News
Preferences
Public Page
Profile Name: My Home Page
Add
Item
Permissions
First Name David Never Ask Always
Middle Init. J. Never Ask Always
Last Name Smith Never Ask Always
Gender Male Female Never Ask Always
Address1 116 Page Mill Road Never Ask Always
Address2 Apt. 300 Never Ask Always
City Palo Alto Never Ask Always
State CA Never Ask Always
Country United States Never Ask Always
Zip Code 94304 Never Ask Always
Homeowner Own Rent Never Ask Always
Click here and we'll tell you what you like!

FIG. 21

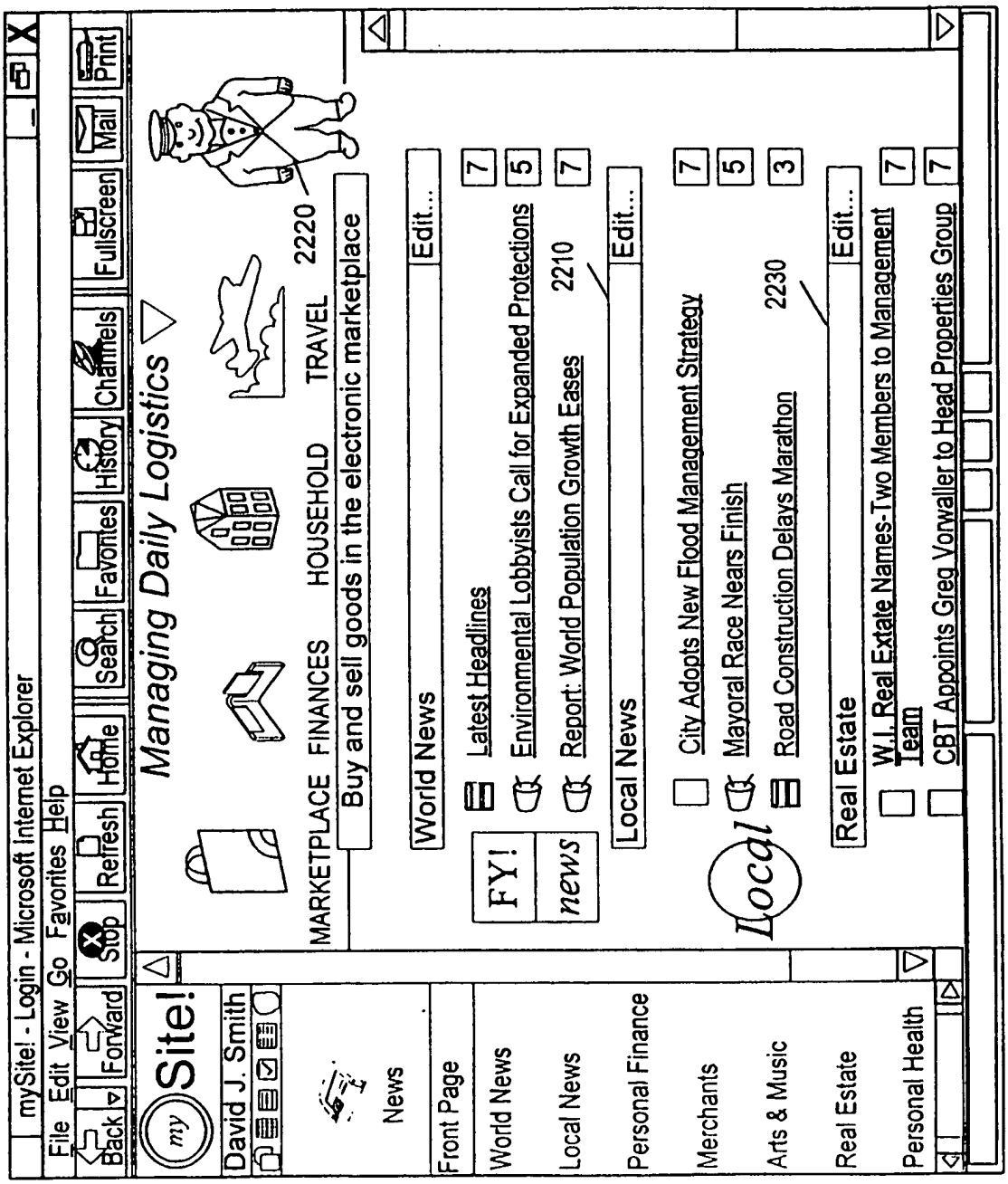

mySite! - Login - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
my Site!
Managing Daily Logistics
David J. Smith
News
MARKETPLACE FINANCES HOUSEHOLD TRAVEL
2220
Buy and sell goods in the electronic marketplace
Front Page
World News
Local News
Personal Finance
Merchants
Arts & Music
Real Estate
Personal Health
World News
Edit...
Latest Headlines 7
Environmental Lobbyists Call for Expanded Protections 5
Report: World Population Growth Eases 7
2210
FY!
news
Local News
Edit...
City Adopts New Flood Management Strategy 7
Mayoral Race Nears Finish 5
Road Construction Delays Marathon 3
2230
Local
Real Estate
Edit...
W.I. Real Estate Names-Two Members to Management Team 7
CBT Appoints Greg Vorwaller to Head Properties Group 7

FIG. 22

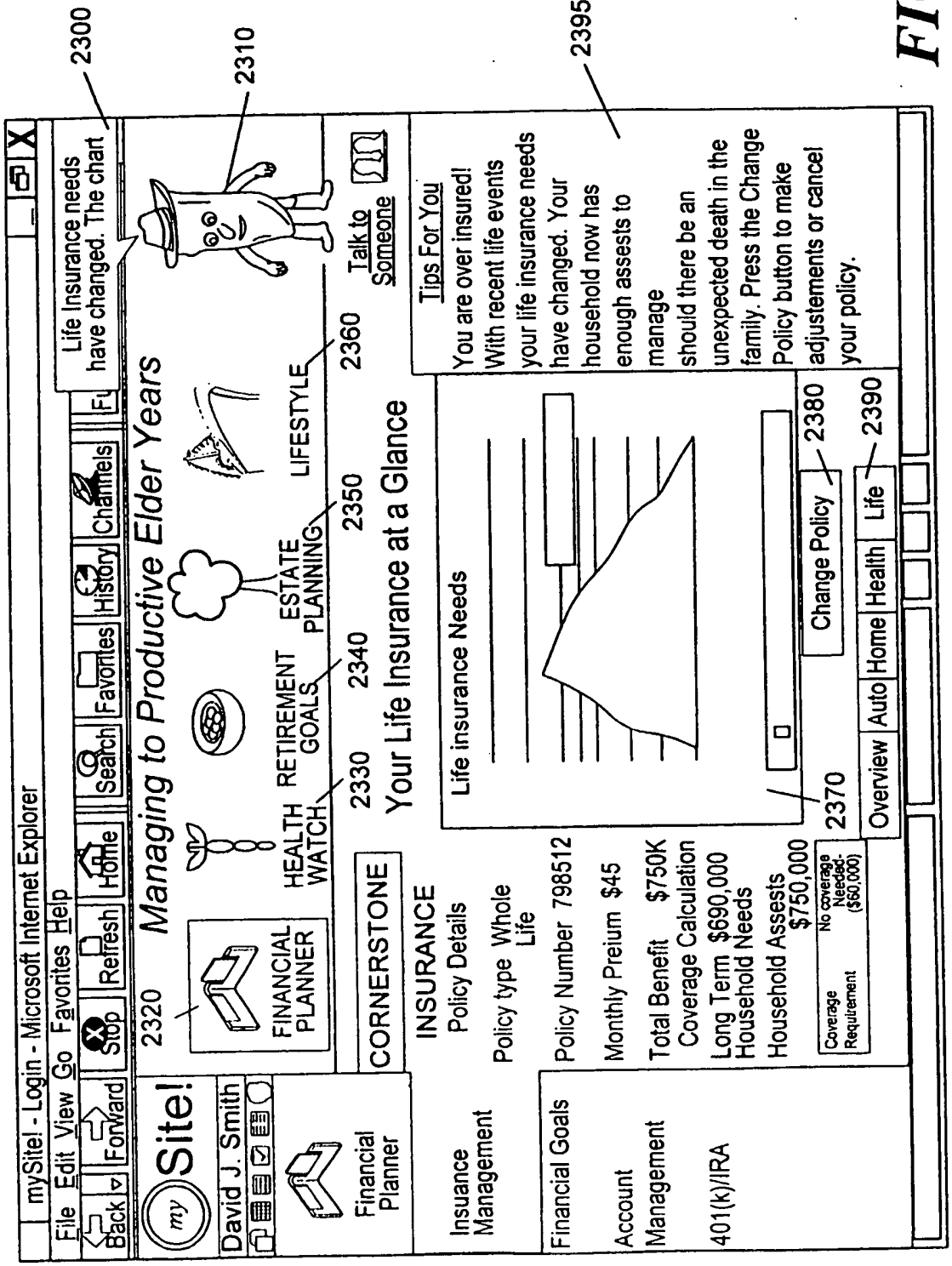

mySite! - Login - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels
my Site!
David J. Smith
Financial Planner
Insuance Management
Financial Goals
Account Management
401(k)/IRA
2320
FINANCIAL PLANNER
CORNERSTONE INSURANCE
Policy Details
Policy type Whole Life
Policy Number 798512
Monthly Preium $45
Total Benefit $750K
Coverage Calculation
Long Term $690,000
Household Needs
Household Assests $750,000
Coverage Requirement
No coverage Needed: ($60,000)
Managing to Productive Elder Years
HEALTH WATCH
RETIREMENT GOALS
ESTATE PLANNING
LIFESTYLE
2330
2340
2350
2360
Your Life Insurance at a Glance
Life insurance Needs
2370
Overview
Auto
Home
Health
Life
2390
Change Policy
2380
Tips For You
You are over insured! With recent life events your life insurance needs have changed. Your household now has enough assests to manage should there be an unexpected death in the family. Press the Change Policy button to make adjustements or cancel your policy.
2395
Talk to Someone
Life Insurance needs have changed. The chart
2300
2310

FIG. 23

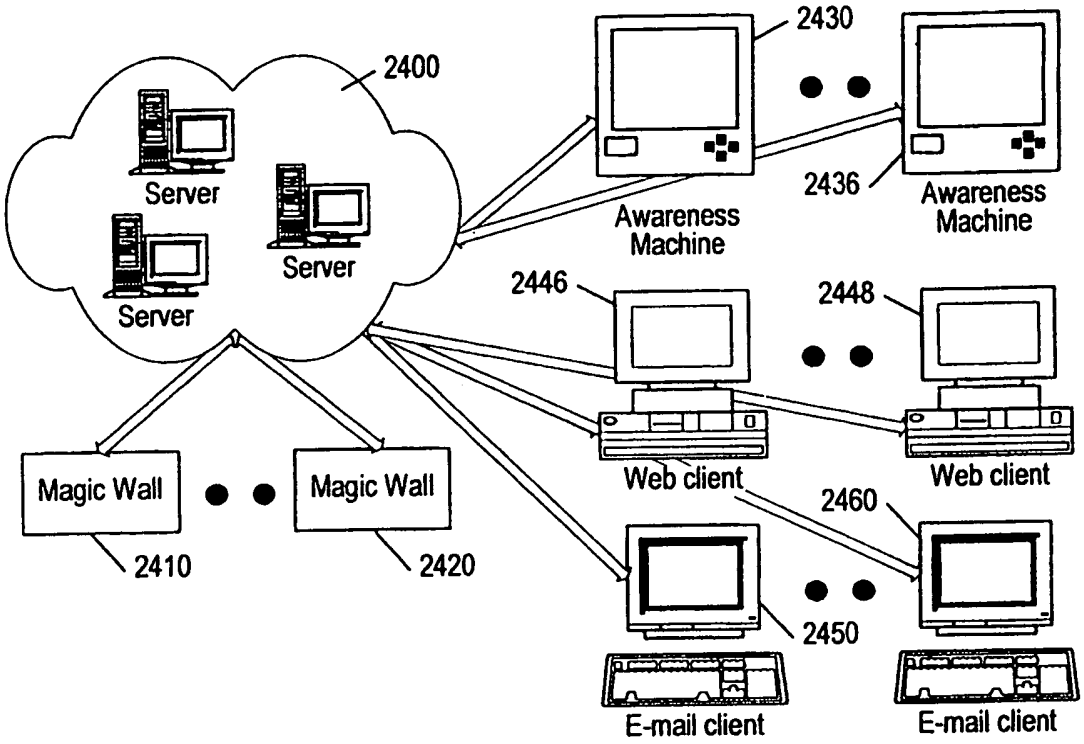

2400
Server
Server
Server
2430
2436
Awareness Machine
Awareness Machine
2446
2448
Web client
Web client
Magic Wall
Magic Wall
2410
2420
2460
2450
E-mail client
E-mail client

FIG. 24

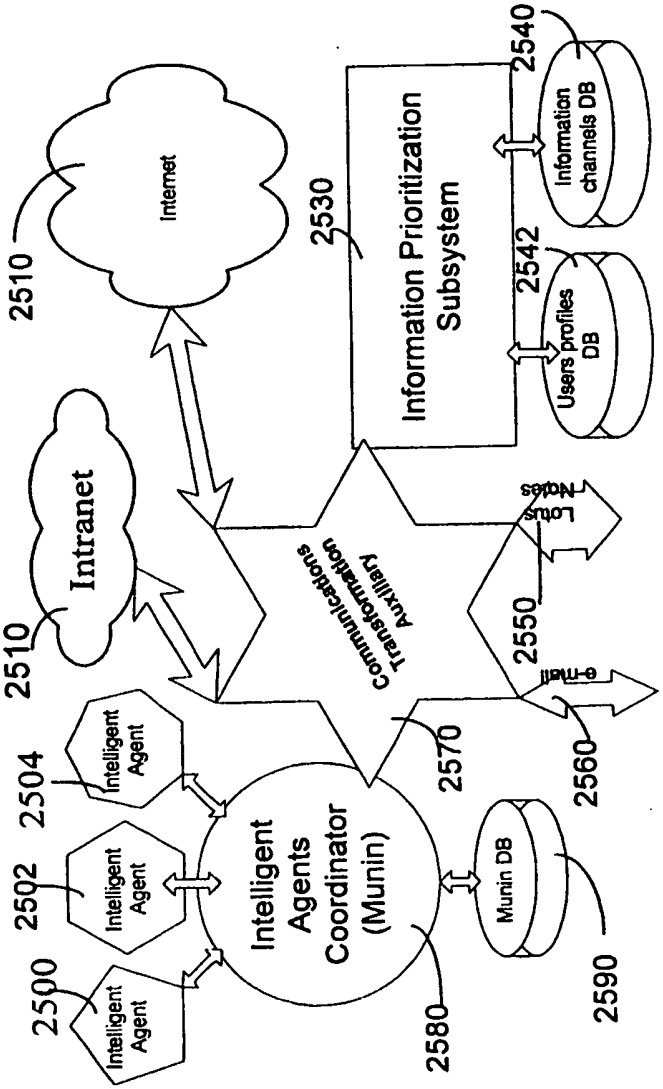

Internet
2510
Intranet
2510
Intelligent Agent
2504
Intelligent Agent
2502
Intelligent Agent
2500
Intelligent Agents Coordinator (Munin)
2580
Munin DB
2590
2570
Communications Transformation Auxiliary
2550
Lotus Notes
e-Mail
2560
Information Prioritization Subsystem
2530
2540
Information channels DB
2542
Users profiles DB

Fig. 25

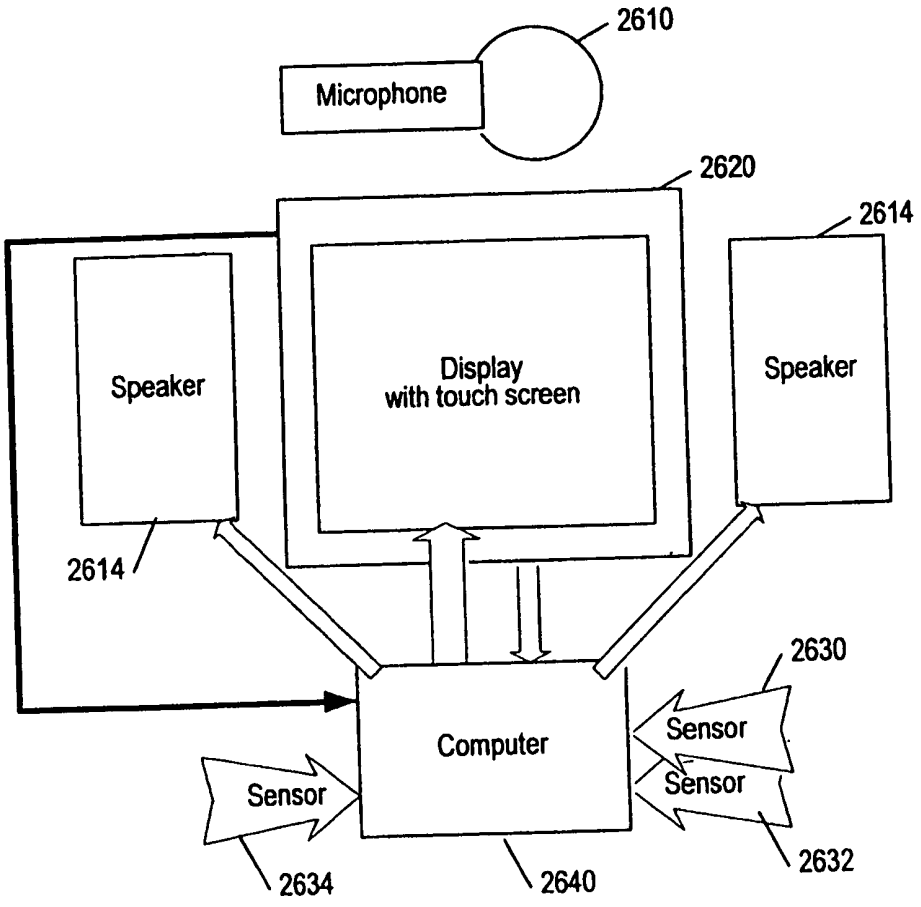

2610
Microphone
2620
2614
Speaker
Display
with touch screen
Speaker
2614
2630
Computer
Sensor
Sensor
Sensor
2634
2640
2632

FIG. 26

IFQ16.2

Recherche

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 367 709 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 89480162.0

(22) Date of filing: 10.10.89

Reg.

(51) Int. Cl.[5] G06F 9/44

(30) Priority: 04.11.88 US 267420

(43) Date of publication of application:
09.05.90 Bulletin 90/19

(84) Designated Contracting States:
DE FR GB IT

(71) Applicant: International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: Kerr, Linda L.
7500 Woodhaven Drive
North Richland Hills Texas 76180(US)

(74) Representative: Tubiana, Max
Compagnie IBM France Département de Propriété Intellectuelle
F-06610 La Gaude(FR)

(54) Customization of user interface for application programs.

(57) An application program automatically creates and presents a customized user interface by determining a set of operations which is appropriate for the current user based on various relevant characteristics of the user and presenting only the specified operations in the menus, icons, application bars or other interface components of the application program.

EP 0 367 709 A1

CUSTOMIZATION OF USER INTERFACE FOR APPLICATION PROGRAMS

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention generally relates to user interfaces to software applications and, more particularly, to the creation, maintenance and use of a user interface to an application program which has been tailored to various characteristics of the current user.

Description of the Prior Art

Computer users vary in many respects, such as job duties, level of familiarity with various application programs, need to use various operations of application programs, frequency of use of various operations of application programs, and right of access to different operations of application programs. Most application programs, however, present a single user interface which does not take into account the individual characteristics of the current user. The result is that the user is presented a user interface which may be confusing, inefficient, insufficiently restricted, or otherwise inappropriate for him.

The user interface for an application program can take many forms, such as menus, sets of icons, sets of action bars, or a command line. The user normally selects an operation or moves through the hierarchy of menus, etc. by some combination of keystrokes, cursor movements, mouse operations, commands or the like.

At any given time, the array of choices presented to the user may be so large as to be confusing, to take up an undue amount of the screen space, or to take up so much memory with the code underlying the choices that system performance is compromised. The array may contain operations which the user does not need or does not yet know how to use. In addition, the user may find it necessary or convenient to access in close sequence operations of a program which appear in different aspects (e.g., menus or action bars) of the user interface. In such a case, it is cumbersome to traverse from menu to menu or from action bar to action bar.

As the user gains familiarity with the application program or as his needs change, he may "outgrow" the user interface, requiring access to a broader or different set of operations.

A number of rudimentary attempts have been made to alter the user interface of an application program, or at least the user's access to aspects of the program or manipulated data, in response to a characteristic of the user. Perhaps the earliest example is the assignment of access levels to database users so that some users may not have write access to certain files or fields in files, some users may not have read access to certain files or fields, and some users may not even be able to access the database at all. This feature is now common among commercially available database management systems.

A more closely related technique is the provision of a software switch, set by the user, which governs whether menus or other explanatory aspects of the user interface are visible on the screen. Thus, a novice user would set the switch to make menus appear on the screen so that he may use the menus as a guide in performing operations. On the other hand, an advanced user would set the switch to remove the menus from the screen so that he could view data in the entire screen, unobstructed by the menu. The word processing program WordStar (trademark of MicroPro) uses such an approach, implemented in the form of differing levels of "help."

An even more recent trend has been to provide so-called "soft interfaces" to application programs, which allow the user to emulate to a degree the interface of a program with which he is already familiar or to create an interface which is purely individual. The word processing program Sprint (trademark of Borland) provides a soft interface which enables the user to operate it with the commands used by WordPerfect (trademark of WordPerfect), WordStar, Word (trademark of Microsoft), certain other word processors, or commands of the user's own selection.

Finally, the methodology exists on a systems level to allow programmers to automate the creation of menus, etc. in the course of application development. The programmer specifies a list of operations, codes, explanatory material, and format parameters, and the menu generator creates a menu for presentation on the screen.

There is nothing in the prior art, however, which enables the user or system administrator to select arbitrarily from the range of operations offered by an application program and thereby create an interface which is truly customized to the level of familiarity, type of job, level of access and other characteristics of the user.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a technique for specifying the range of operations to be presented by the user interface of an application program which is customized for particular users or classes of users and is readily changeable by the user or system administrator.

It is another object of the invention to provide a program for creating and displaying the relevant, customized user interface which has been specified by the user or system administrator.

According to the invention, the user interface of an application program is restructured in a form customized for the current user each time he invokes the program. The user or system administrator first creates a customization table which specifies the operations to be made available to particular classes of users or even individual users. For example, a matrix could be created which specifies certain groups of operations for presentation to users depending on the users' type of job and degree of familiarity with the application program. The customization table could be modified whenever desired. Each time a user invokes the application program, the appropriate set of operations for the user is determined from the customization table by searching a user profile table based on the user's identifier, prompting the user to enter his relevant characteristics, or other means.

Because the user's level of familiarity with the application program will frequently be the most important characteristic, a count of the number of times each user has invoked the application program could be stored and used to estimate the user's level of familiarity and select the appropriate set of operations for the user, automatically raising the level of user interface as thresholds are reached. The user could be permitted to override the system count if it led to an inappropriate level of interface.

Similar types of automatic upgrading could occur by monitoring the number of serious errors that occur in each session and raising the complexity of the user interface if fewer than a certain number occur during the session, or by monitoring calls to operations not presented by the interface and adding an operation to the user interface if it is called more than a certain number of times during the session.

The set of operations appropriate to the user is then used to generate menus, icons, action bars or other interface components which present only the specified operations, rather than the full, and possibly confusing, inefficient or otherwise inappropriate, range of operations available within the application program.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, aspects and advantages of the invention will be better understood from the following detailed description of a preferred embodiment of the invention with reference to the drawings, in which:

FIG. 1 illustrates a typical table specifying the operations that might be made available to various types of users with various levels of experience with the application program in the practice of the invention;

FIG. 2A illustrates a typical menu which may be presented to a novice clerical user in the practice of the invention;

FIG. 2B illustrates a typical menu which may be presented to a moderately experienced clerical user in the practice of the invention;

FIGS. 3 to 12, taken together, constitute a flow chart of the user interface customization procedure according to the invention in which the connectors in each of the figures indicate the manner in which the several figures are interconnected to form the flow chart and wherein

FIG. 3 is a portion of the flow chart which illustrates customization table creation and editing logic;

FIG. 4 is a portion of the flow chart which illustrates user profile table creation and editing logic;

FIG. 5 is a portion of the flow chart which illustrates initialization and user lookup logic;

FIG. 6 is a further portion of the flow chart which illustrates operation set creation and interface generation logic;

FIG. 7 is a further portion of the flow chart which illustrates exception monitor and operation monitor invocation logic;

FIG. 8 is a further portion of the flow chart which illustrates exit testing and upgrading logic based on the contents of the operation monitor list and the total in the operation count;

FIG. 9 is a further portion of the flow chart which illustrates exit testing and upgrading logic based on the total in the exception count;

FIG. 10 is a further portion of the flow chart which illustrates exit testing and upgrading logic based on the user's desires;

FIG. 11 is a further portion of the flow chart which illustrates exception monitor logic; and

FIG. 12 is a further portion of the flow chart which illustrates operation monitor logic.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT OF THE INVENTION

Referring now to the drawings, and more particularly to FIG. 1, the procedure according to the invention creates, stores in electronic storage, and utilizes a customization table which specifies sets of operations to be presented in the user interface to users with the corresponding relevant characteristics. FIG. 1 illustrates a customization table which considers only two user characteristics, namely the familiarity level characteristic 10 and the job type characteristic 12. It will be observed that the appropriate operations set for a user with clerical job type characteristic 14 and novice familiarity level characteristic 16 is operations set 18, comprising the operations listed therein.

FIG. 2 illustrates two sample user interface menus for a user with clerical job type. FIG. 2A illustrates that note function 20, file subfunction 22 provides the novice familiarity level clerical job type user with operations set 24. In contrast, FIG. 2B illustrates that the moderate familiarity level clerical job type user is provided with operations set 26, which contains the opera tions Link, Merge, and Page Setup in addition to the operations shown in operations set 24 of FIG. 2A.

FIGS. 3 and 4 illustrate the logic of the programs which create and edit the customization table and the user profile table, both of which are used to generate customized user interfaces for individual users. The customization table and user profile table are searchable aggregations, such as arrays, of multifield records as are commonly used in computer programming applications and may be created by a variety of known methods such as interactive entry or editing using a text editor, database management system or set of table creation functions embedded within a table maintenance routine specialized to the present invention.

FIG. 3 illustrates the program which creates and edits the customization table. When this program is first invoked, a test is made to determine whether the customization table exists, as indicated in decision block 30. If the table exists, it is loaded into memory as indicated in block 31. If the table does not exist, a blank table format, which contains table characteristics such as field names, field lengths and required data types, is loaded into memory as indicated in block 32. The existing table or the blank table format, as the case may be, is then displayed to the user as indicated in block 33. The program then enters edit mode so that the user may add or delete records and modify fields in the table as indicated in block 34. Editing ends when the user enters the quit editing command as indicated in block 35. The newly created or edited table is then stored in external memory as indicated in block 36.

FIG. 4 illustrates the program which creates and edits the user profile table. When this program is first invoked, a test is made to determine whether the user profile table exists, as indicated in decision block 40. If the table exists, it is loaded into memory as indicated in block 41. If the table does not exist, a blank table format, which contains table characteristics such as field names, field lengths and required data types, is loaded into memory as indicated in block 42. The existing table or the blank table format, as the case may be, is then displayed to the user as indicated in block 43. The program then enters edit mode so that the user may add or delete records and modify fields in the table as indicated in block 44. Editing ends when the user enters the quit editing command as indicated in block 45. The newly created or edited table is then stored in external memory as indicated in block 46.

FIG. 5 illustrates that the automatic customization procedure according to the invention begins by determining the user's identifier in block 50 by known methods such as testing the system variable which contains the current user's identifier or by prompting the user to enter his or her identifier. The user profile table is then loaded into memory as indicated in block 51, and the user profile table is searched for the current user's identifier in decision block 52. If the user's identifier is not found in the user profile table, the user is prompted to enter data into the fields of a blank user profile record which has been created for that particular user, as indicated in block 53. The data input by the user is moved into the appropriate fields to create the user profile record, which is retained in memory for further use, as indicated in block 54. The newly created user profile record is also added to the user profile table as indicated in block 55, and the program moves to block 60 shown in FIG. 6.

If, on the other hand, the user identifier is found in the user profile table, the user's profile record is moved from the table into memory as indicated in block 56, and the program moves to block 60 of FIG. 6.

In block 60 of FIG. 6, a multikey search is carried out in the customization table to find the entry corresponding to the user characteristics specified in the user profile record. Typical user characteristics contained in the user profile record might include the user's job type, level of familiarity with the application program, number of times the user has invoked the application program, data access and modification rights, and others. There is one field of the user profile record which, if it exists, will not be used in the search. This is a field which lists operations individually specified to be added to the user's customized interface in addition to the operations that would be included based solely on the characteristics in the user profile

record. The purpose of this is to provide for an arbitrary degree of customization beyond that provided by the generalized sets of operations deemed appropriate to users of various types.

The list of operations which has thus been generated from both the customization table and the individualized operations of the user profile record is passed to the interface generator as indicated in block 61. The interface generator then creates the corresponding set of interface elements, such as menus, application bars, etc., by merging the operations in the list into the appropriate interface screen designs as indicated in block 62.

In FIG. 7, the exception monitor is invoked by setting the exception count to zero as indicated in block 70 and setting a switch to respond to exception interrupts by calling the exception monitor program as indicated in block 71. The operation monitor is then invoked by emptying the operation list as indicated in block 72, setting the operation count to 0 as indicated in block 73, and setting a switch to call the operation monitor program any time an operation is invoked as indicated in block 74. Control is then passed to the application program for presentation of the top level interface element (e.g., main menu) and use of the application program by the user as indicated in block 75.

FIG. 8 illustrates the exit routine which is entered after the user exits the application program as indicated in block 80. The operation list is loaded from the opera tion monitor as indicated in block 81, and the operations which were invoked more than a predetermined number of times x during the session are displayed to the user as indicated in block 82. The user is then asked if he wishes to add the displayed operations to his customized user interface as indicated in decision block 83. If the user answers in the affirmative, the new operations are added to the individualized operation field of the user's user profile record as indicated in block 84 and the program moves to decision block 85. If the user answers in the negative, the program moves directly to decision block 85.

In decision block 85, the operation count is tested to determined whether it contains more than a predetermined number y of entries. If so, the field for the user's experience level is incremented in the user profile record as indicated in block 100 of FIG. 10 and the program moves to block 101. If not, the program moves to decision block 90 of FIG. 9.

In FIG. 9, the exception count is tested to determine whether the total number of serious errors is less than a predetermined number z as indicated in decision block 90. If so, the field for the user's experience level is incremented in the user profile record as indicated in block 100 of FIG. 10 and the program moves to block 101. If not, the program moves to block 91 of FIG. 9.

The experience count field, which stores the number of times the user has accessed the application program, is incremented in the user profile record as indicated in block 91, and the incremented value is then tested as indicated in decision block 92 to determine whether the value is greater than a predetermined number w. If so, the field for the user's experience level is incremented in the user profile record as indicated in block 100 of FIG. 10 and the program moves to block 101. If not, the program moves directly to block 101.

In FIG. 10, a message is displayed to the user as indicated in block 101 which lists the upgrading actions which have been taken or states that no upgrading actions have been taken, as the case may be. The user is then asked in decision block 102 whether he wishes to add to, delete or change the upgrading actions. If so, the upgrade is modified in accordance with user input as indicated in block 103, and the program terminates. If not, the program terminates directly.

FIG. 11 illustrates the operation of the exception monitor. Whenever an exception interrupt is generated as indicated in block 110, it is tested as indicated in decision block 111 to determine whether the severity of the error exceeds a predetermined level. If so, the exception count is incremented as indicated in block 112 and control is returned to the main program. If not, control returns directly to the main program.

FIG. 12 illustrates the operation of the operation monitor. Whenever an operation is invoked as indicated in block 120, it is tested as indicated in decision block 121 to determine whether the operation is already contained in the user's customized interface. If not, the operation is added to the operation list maintained by the operation monitor as indicated in block 122, the operation count is incremented as indicated in block 123, and control is returned to the main program. If so, control returns directly to the main program.

While the invention has been described in terms of a single preferred embodiment, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the appended claims.

Claims

1. In a data processing system, a procedure for automatically customizing a user interface of an application program to present a limited set of operations based on relevant characteristics of a user comprising the steps of:

accepting input data relating to identifiers of actual or potential users of the application program, relevant characteristics of users and/or classes of users' and sets of operations of the application program which are appropriate for each such user or class of users;
creating a user profile table which stores said relevant characteristics of users in a format which is searchable using the identifier of a user as a key value;
storing the user profile table for later access;
creating a customization table which stores said sets of operations in a format which is searchable using said relevant characteristics as key values;
storing the customization table for later access;
determining user identifier or relevant characteristics of a user when a user invokes the application program;
searching the customization table for the appropriate set of operations based on the identifier or relevant characteristics of the user; and
automatically generating user interface components during the running of the application program which present only the operations specified for the user.

2. The procedure for automatically customizing a user interface of an application program as recited in claim 1 wherein the step of determining relevant characteristics of a user is performed automatically by using an identifier of the user as a key to search said user profile table.

3. The procedure for automatically customizing a user interface of an application program as recited in claim 1 further comprising the steps of:
maintaining a count of the number of times a user has invoked the application program;
determining said user's level of familiarity with the application program by reference to said count; and
utilizing said level of familiarity with the application program as one of the relevant characteristics of the user in searching the customization table.

4. The procedure for automatically customizing a user interface of an application program with system determination of the user's level of familiarity with the application program as recited in claim 3 further comprising the step of providing the user an option to override the count of times the user has invoked the application program to allow for variations in the time different users take to familiarize themselves with the application program.

5. The procedure for automatically customizing the user interface of an application program as recited in claim 1 further comprising the steps of:
maintaining a count of the number of serious exception conditions which occur during a user's session with the application program;
automatically increasing said user's level of familiarity with the application program each time a session ends with said count being lower than a predetermined number; and
storing said increased level of familiarity with the application program in the user profile table for searching the customization table in subsequent sessions by the same user with the application program.

6. The procedure for automatically customizing the user interface of an application program with automatic in creasing of a user's level of familiarity with the application program as recited in claim 5 further comprising the step of providing the user an option to override said automatic increase in a user's level of familiarity with the application program.

7. The procedure for automatically customizing the user interface of an application program as recited in claim 1 further comprising the steps of:
maintaining a list of operations not in a user's customized operations set which are invoked by the user during a session with the application program;
automatically adding to the user's customized operations set all operations from said list which were invoked more than a predetermined number of times within the session; and
utilizing said additional operations in creating the user's customized user interface in subsequent sessions by the same user with the application program.

8. The procedure for automatically customizing a user interface of an application program with automatic adding of certain operations to a user's customized operations set as recited in claim 7 further comprising the step of providing the user an option to override said automatic addition of operations to the user's customized operations set.

9. The procedure for automatically customizing the user interface of an application program as recited in claim 1 further comprising the steps of:
maintaining a count of operations not in a user's customized operations set which are invoked by the user during a session with the application program;
automatically increasing said user's level of familiarity with the application program each time a session ends with said count being higher than a predetermined number; and
storing said increased level of familiarity with the application program in the user profile table for searching the customization table in subsequent sessions by the same user with the application program.

10. The procedure for automatically customizing the user interface of an application program with automatic increasing of the user's level of familiarity with the application program as recited in claim 9 further comprising the step of providing the

user an option to override said automatic increase in the user's level of familiarity with the application program.

FIG. 1

SAMPLE
CUSTOMIZATION TABLE

JOB TYPE (12)	FAMILIARITY LEVEL (10)		
	NOVICE (16)	MODERATE	FREQUENT
CLERICAL (14)	CALENDAR LIMITED NOTE LIMITED EDIT MAIL MESSAGE REMINDER FILE (18)	CALENDAR EXPANDED NOTE EXPANDED EDIT MAIL MESSAGE REMINDER TEXT DOCUMENTS EXPANDED FILE	CALENDAR EXPANDED NOTE EXPANDED EDIT MAIL MESSAGE REMINDER COMPOSITE DOCUMENTS GRAPHICS EXPANDED FILE
EXECUTIVE	---	---	---
PROFESSIONAL / TECHNICAL	---	---	---
SCIENTIFIC/ ENGINEERING	---	---	---

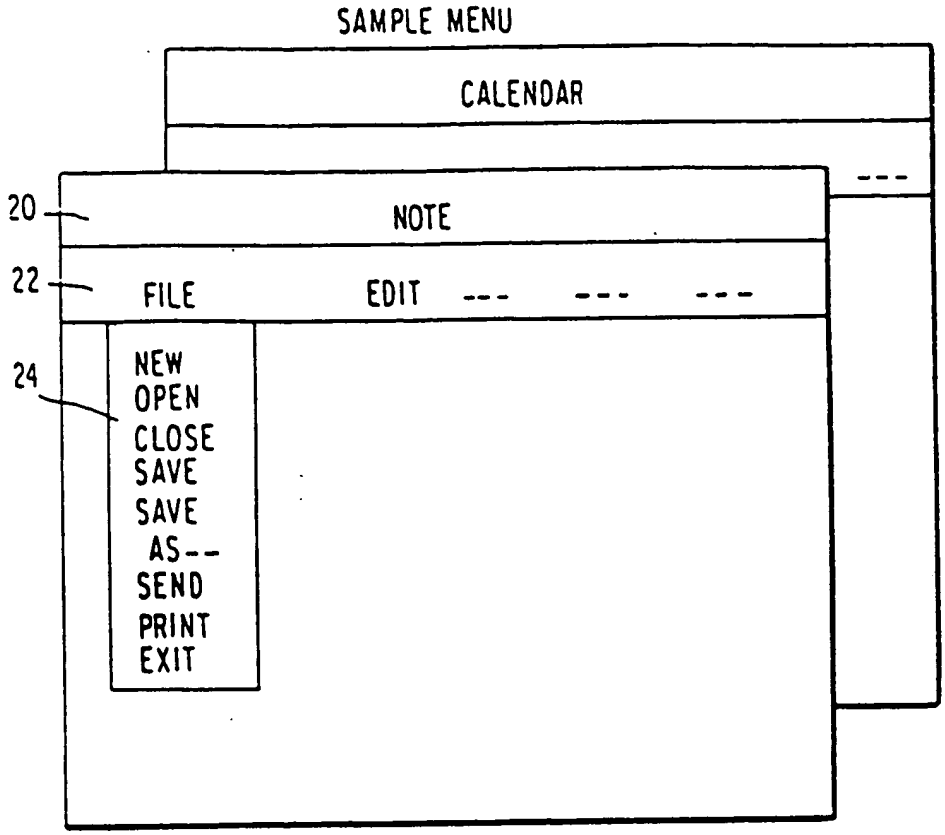


FIG.2A
"NEW" LEVEL
CLERICAL

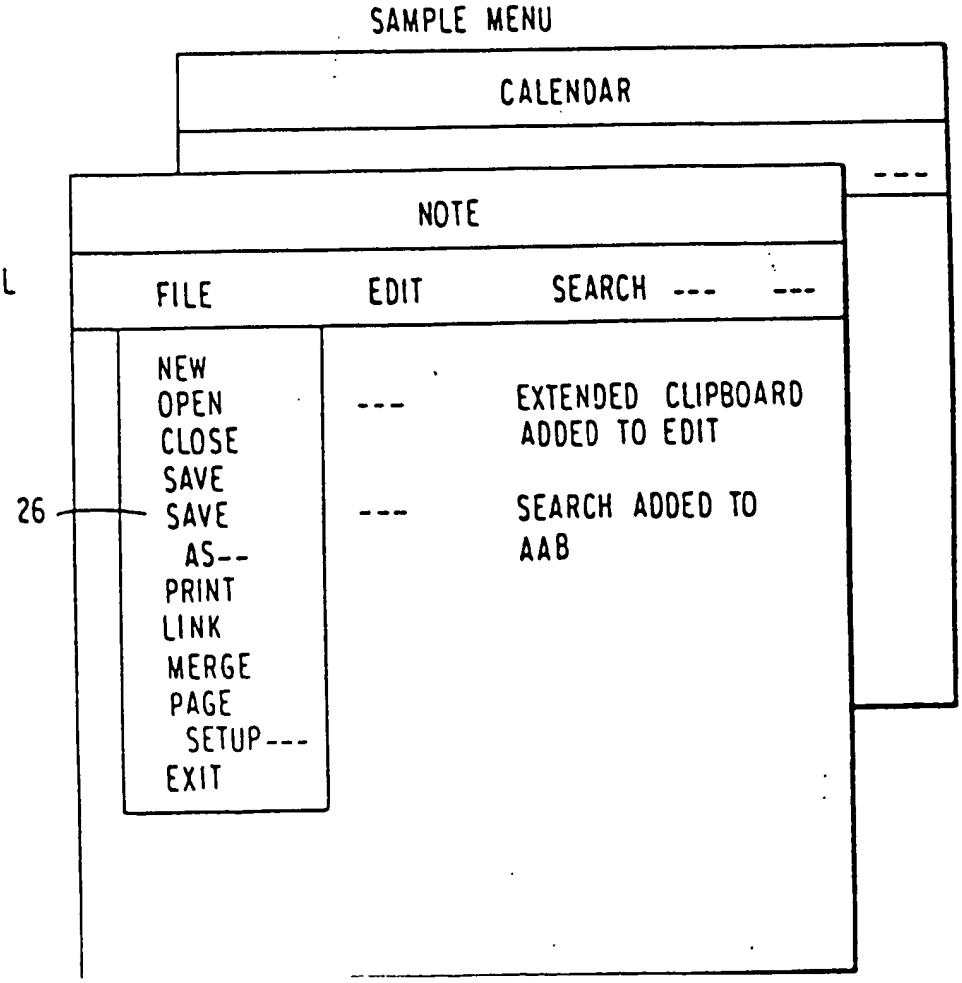


FIG.2B
"MODERATE" LEVEL
CLERICAL

FIG.3

PROGRAM TO CREATE AND EDIT CUSTOMIZATION TABLE

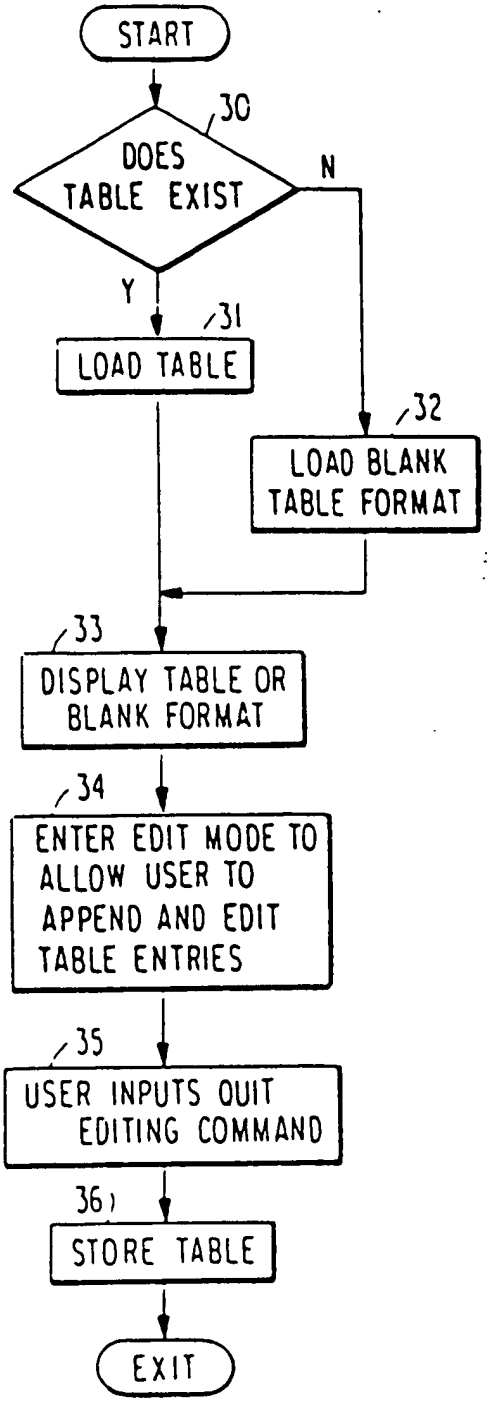


FIG.4

PROGRAM TO CREATE AND EDIT USER PROFILE TABLE

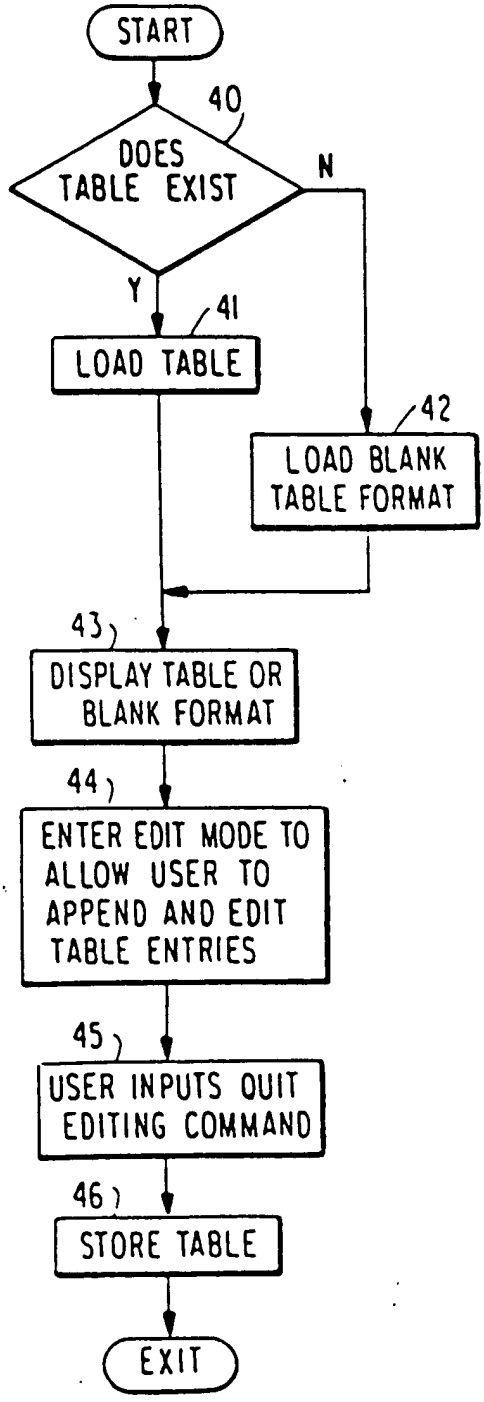


FIG.5

AUTOMATIC USER INTERFACE CUSTOMIZATION PROCEDURE

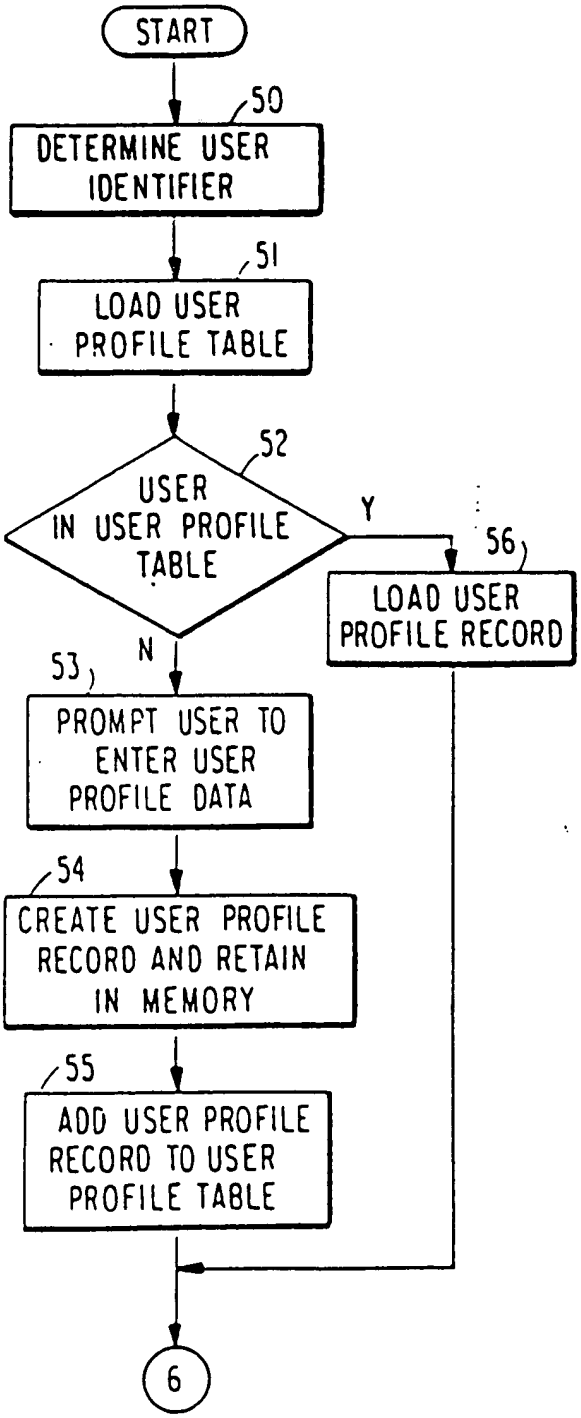
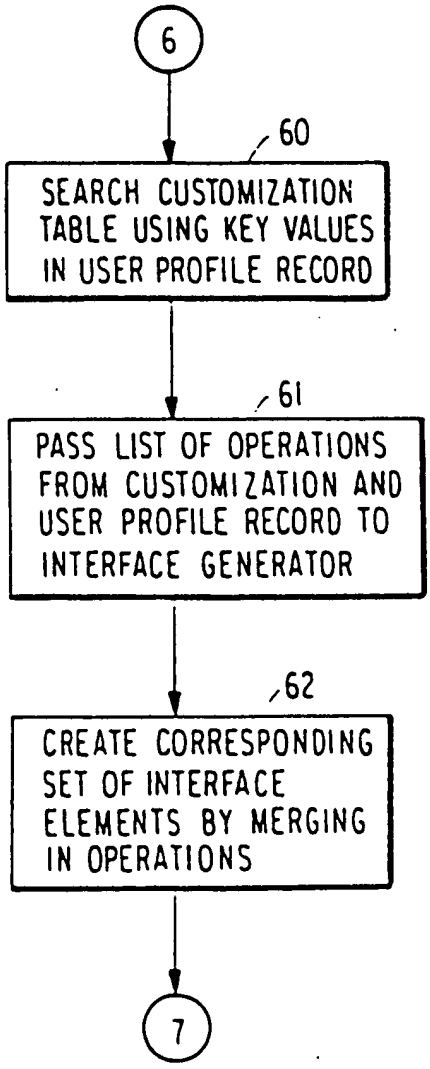


FIG.6



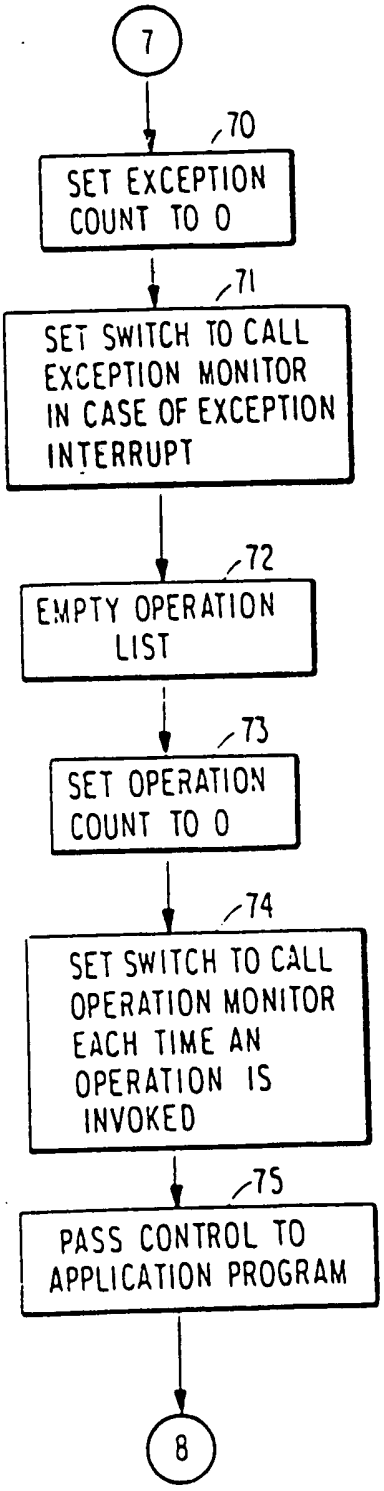

7
70
SET EXCEPTION COUNT TO 0
71
SET SWITCH TO CALL EXCEPTION MONITOR IN CASE OF EXCEPTION INTERRUPT
72
EMPTY OPERATION LIST
73
SET OPERATION COUNT TO 0
74
SET SWITCH TO CALL OPERATION MONITOR EACH TIME AN OPERATION IS INVOKED
75
PASS CONTROL TO APPLICATION PROGRAM
8

FIG. 7

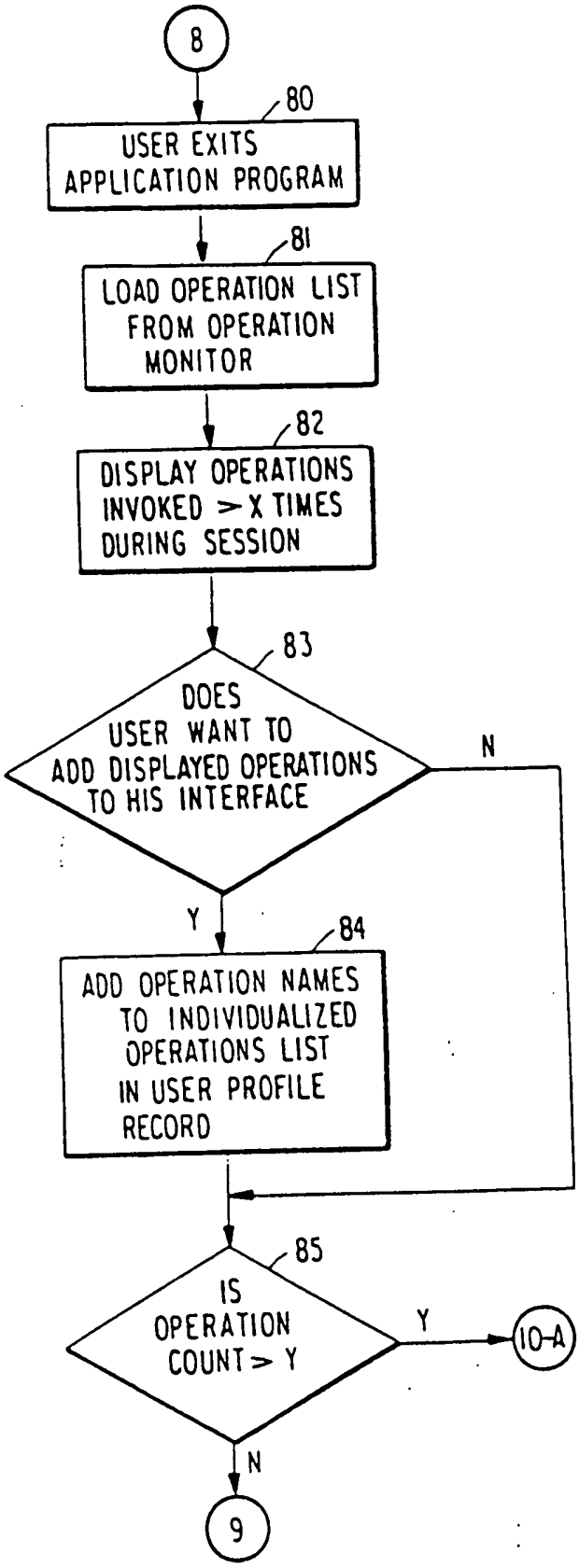

8
80
USER EXITS APPLICATION PROGRAM
81
LOAD OPERATION LIST FROM OPERATION MONITOR
82
DISPLAY OPERATIONS INVOKED > X TIMES DURING SESSION
83
DOES USER WANT TO ADD DISPLAYED OPERATIONS TO HIS INTERFACE
N
Y
84
ADD OPERATION NAMES TO INDIVIDUALIZED OPERATIONS LIST IN USER PROFILE RECORD
85
IS OPERATION COUNT > Y
Y
10-A
N
9

FIG. 8

FIG.9

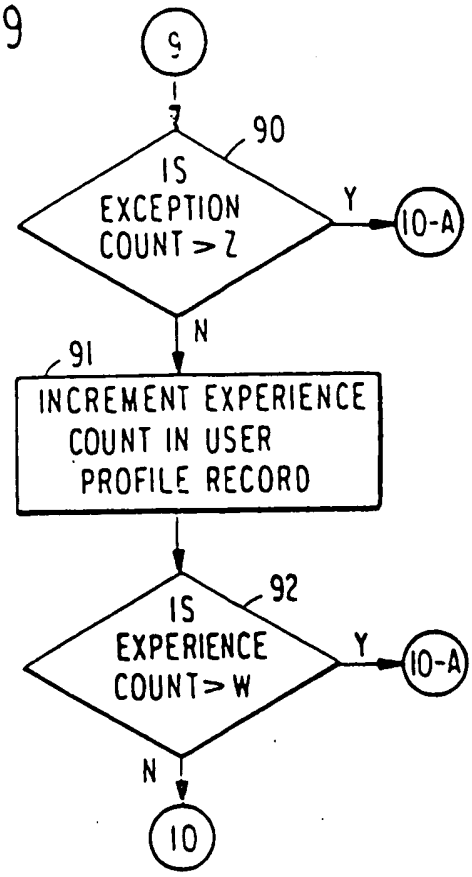
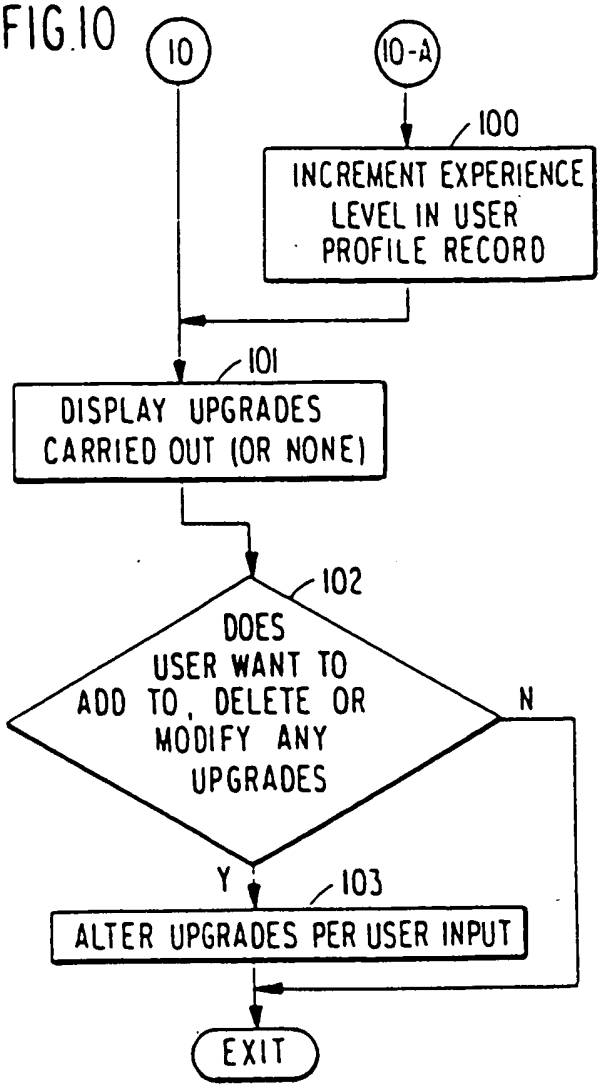


FIG.10



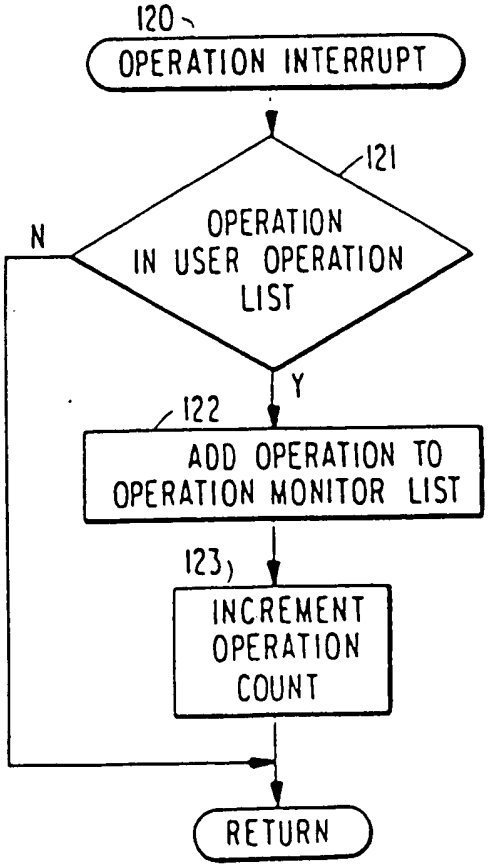


FIG.12

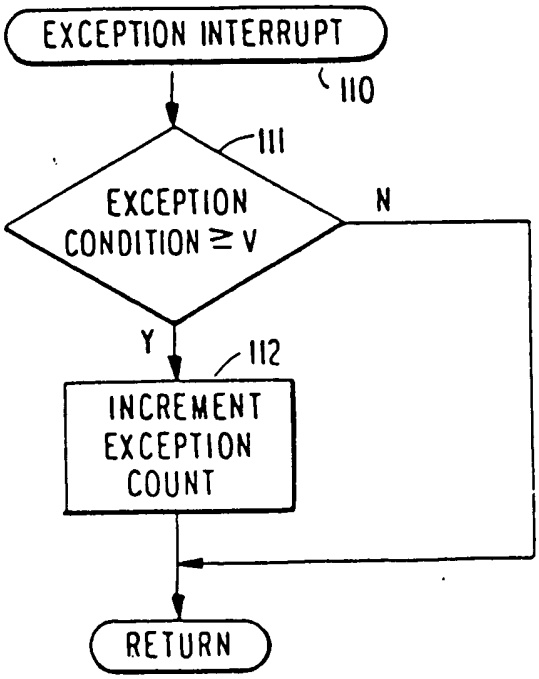


FIG.11

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 89 48 0162

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
X	SYSTEMS & COMPUTERS IN JAPAN, vol. 19, no. 10, October 1988, pages 80-86, Scripta Technica, Inc., Silver Spring, MD, US; M. NAGATA et al.: "A method for generating messages of the interactive software based on the individual user model" translated from JOHO TUSHIN GAKKI RONBUNSHI, vol. 70-D, no. 11, November 1987, pages 2077-2082; * Page 81, right-hand column, line 50 - page 82, left-hand column, line 23; page 82, left-hand column, line 1 - page 84, right-hand column, line 4 *	1-3,5	G 06 F 9/44
A	INFORMATION PROCESSING '86, PROCEEDINGS OF THE IFIP 10TH WORLD COMPUTER CONGRESS, Dublin, 1st - 5th September 1986, pages 481-484, IFIP; D.D. NOVATCHEV: "Future interface management systems - successors of modern UIMS?" * Page 483, right-hand column, lines 19-47 *	1	TECHNICAL FIELDS SEARCHED (Int. Cl.5)
A	"Distributed artificial intelligence", edited by M.N. HUHNS, 1987, Pitman, London, GB; * Chapter 9: "DAI for document retrieval: The MINDS project"; page 249, lines 1-11 *	1,4,6-9	G 06 F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	29-01-1990	GRASLAND B.Y.R.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

PO FORM 1503 03.82 (P0401)

CAD Info.Net

Macro Express 2000, The Smart Way to Compute

Press Release

Macro Express 2000, The Smart Way to Compute

Bountiful, UT - August 24, 1999 - Insight Software Solutions, Inc. today announced the release of Macro Express 2000, the intelligent solution for repetitive and tedious computer tasks.

Macro Express 2000 adds a number of new features such as text, integer and decimal variables, if / then / else logic and the processing of text files or folders. Other notable new features include logging messages to files, menu or question prompting, password protection, macro activation with floating menus or mouse, and more.

"We anticipate most of the excitement to be generated from the new variables and logic functions," said Michael D. Jones, President of Insight Software Solutions, Inc. "These functions open up a whole new realm of macro possibilities. Users can now create macros to pretty much do the thinking for them."

Variables allow input of data into the macro from a variety of sources, such as from the macro itself, a file, user input, clipboard, INI files, a window title or others. The data can then be manipulated, processed and tested. Actions can be taken on the results of the test or on the variables themselves. For example, the new Process ASCII Delimited Text File command makes use of variables by inserting each field of a record into a variable. The variable can then be used to paste the data into other applications.

The If / Then / Else capabilities allow users to test for a variety of conditions and choose actions based on the result of the condition. Conditions may include the existence of a file or folder, if a window or program is running or on top, if the clipboard contains certain text, if a variable equals, is less than, etc. to another variable or value, the response to a question or a menu and others. For example, test if a program exists. If it does, run the program. If it doesn't (the else part), display a message indicating the program could not be found.

Macro Express 2000 provides two new methods for quick activation of macros. A floating menu of macros remains always on top and always accessible as long as Macro Express is running. Just click on one of the macros in the floating menu to activate that macro for playback. Users can also trigger macros with the new mouse click feature. Assign macros to activate by clicking on specific areas of the screen or by clicking on various parts of a window, such as the scroll bar, title bar, etc.

For enhanced privacy and security, users can apply the new password protection command that requires a password be entered before the macro will continue. In addition, a macro that contains this command cannot be edited unless the password is first entered. This prevents someone from simply looking at the macro contents or deleting the password command and then playing the macro to get at sensitive information. The encrypted text command gives additional security for sensitive information.

The new Log Errors command logs any errors to a file. An entry is made in the error log file as soon as the command is processed. A macro completion entry is also made to the log file and all entries are date/time stamped. Similarly, the new Log Message feature logs messages to a file. Users can specify the file name or log it to the default error log file.

The new Menu command allows a macro to display a menu of choices. Choices can be displayed as either radio boxes (one selection out of the group) or as checkboxes (select as many as wanted). Up to 10 menu choices can be included along with header text describing the purpose of the menu.

Macro Express allows users to automate all of their common computer tasks. Dozens of wizards are available for step by step macro creation. Wizards include typing text, symbols and keystrokes, pasting text, capturing macros, setting networking connections, launching web sites, downloading files, creating reminders, resizing, moving, maximizing windows, selecting printers and screen resolutions and many more. Record, edit (via an easy to use scripting editor) and play back macros. Assign macros for global use or for a specific program or window. Launch macros with a Hot Key, ShortKey, Popup Menu, Mouse Click, Floating Menu, Window Title or via a Timed Schedule.

Macro Express 2000 will be featured exclusively on ZDNET's hotfiles.com beginning August 24, 1999. A 30 day trial version will be available for download. The program will also be available from our web site at www.macros.com. Macro Express includes an extensive help file, a tutorial (in pdf format) and numerous example macros.

Macro Express is suitable for use with Windows 95, 98 or NT and is priced at $34.95 + $3.00 S&H in the USA. Shipping charges are waived if ordered for electronic delivery from our web site. Version 2.0 has an upgrade price of $15.00 for existing users. The program may be obtained directly from Insight Software Solutions, Inc., P.O. Box 354, Bountiful, UT 84011-0354, USA, Tel: (801) 295-1890, Fax: (801) 299-1781, E-Mail: info@wintools.com.